

COMMODORE 64™
Book/Software

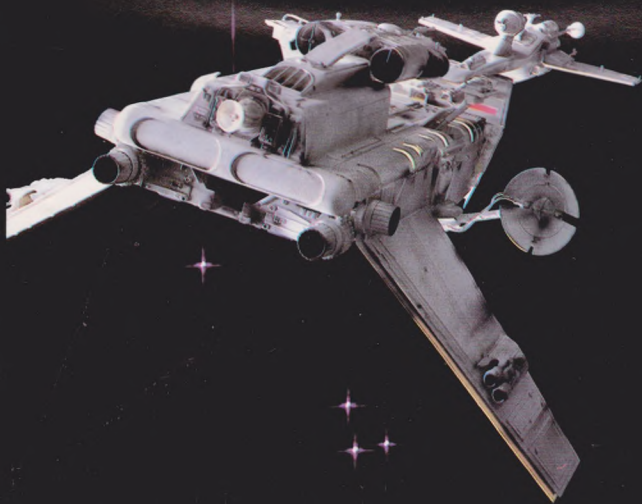
Lost in Space

1

Programming Adventures in BASIC

Lost in Space

Commodore 64™ version



HAYDEN
7633-9

HAYDEN Book/Software

A revolutionary FUN approach to computer learning

- *Race against time to escape from an alien planet by applying your understanding of BASIC to control a spaceship computer.*
- *Discover the wonders of BASIC programming using your Commodore 64.*

You are the hero of this mission in space. By some strange accident you are transported to a faraway galaxy and join three wanderers. There is Dr. W—a wise, but slightly odd, teacher and inventor; Aleat—a beautiful 240-year-old Arfonite; and her trusty, rusty robot companion SN7. Aboard a spaceship long ago deserted by its crew, you find a computer remarkably similar to your Commodore 64. If you can make the computer understand your instructions you can escape. If you can't operate the computer, you'll remain LOST IN SPACE.

The *software* inside this package is specially designed to teach the fundamentals of BASIC programming, and it contains programs from the spaceship computer.

Dr. W also carefully explains each BASIC command in an easy-to-understand section at the end of the book.



HAYDEN BOOK COMPANY

a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

Lost in Space

Commodore 64™ version

HAYDEN book/software

LIMITED WARRANTY

This disk and its contents are provided only as a convenience for program or data entry. The

Publisher, Hayden Book Company, Inc., warrants to the original purchaser only that the programs contained in this package are substantially the same as those contained in the corresponding Hayden book, and that these programs will load and run on the manufacturer's standard configuration of the computer equipment listed.

The Publisher makes no other warranty of any kind, express or implied, including but not limited to warranties of merchantability or fitness for any particular use. Neither the Publisher nor the author(s) is liable to any purchaser or user for any damage or loss (including loss of business or profits) caused directly or indirectly by the use of, or inability to use, this disk, documentation, or other contents of this package.

The Publisher will replace a defective disk without further charge if it is returned within 10 days of purchase to Hayden Book Company, Inc., at the address listed. The returned disk must be in its original envelope and must be accompanied by proof of purchase. This warranty gives you specific legal rights and you may have other rights that vary from state to state.

**For extended media life—
here's how to take care of your flexible disk**



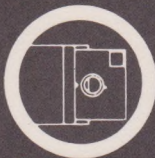
Precision surface.
No fingers, please!



For your disk's sake
(and the system's, too)
insert disk carefully.



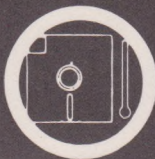
Magnetic fields erase.
Keep them far away.



Keep it safe—
in the envelope
when not in use.



Bending and folding
may damage.
Handle with care.



Keep disks comfortable
Store at 10° to 50°C
50° to 125° F

1

Programming Adventures in BASIC

Lost in Space

Commodore 64™ version



HAYDEN Book/Software



DR. WATSON COMPUTER LEARNING SERIES



Lost in Space

Commodore 64™ version

Glentop/Honeyfold



HAYDEN BOOK COMPANY

a division of Hayden Publishing Company, Inc.

Hasbrouck Heights, New Jersey

All programs in this book and the accompanying software have been written expressly to illustrate specific teaching points. They are not warranted as being suitable for any particular application. Every care has been taken in the writing and presentation of this book but no responsibility is assumed by the author or publisher for any errors or omissions contained herein.

The Story: TOM DeHAVAS

Dr W Explains: MARTIN E. THOMPSON

Developmental Editor: KAREN PASTUZYK

Production Editor: TERRY DONOVAN

Illustrations: NETWORK GRAPHICS, LANE YERKES/NIK STANBURY

Cover Designs: JIM BERNARD/JOHN M-RÖBLIN

Printed and bound by: GENERAL OFFSET COMPANY, INC.

Commodore 64 is a trademark of Commodore Business Machines, Inc., and Dr. Watson is a trademark of Glentop Publishers Ltd., both of which are not affiliated with Hayden Book Company.

Copyright © 1983, 1984 by Glentop Publishers Ltd. All rights reserved. No part of this book may be reprinted, or reproduced, or utilized in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage and retrieval system, without permission in writing from the Publisher.

1	2	3	4	5	6	7	8	9	PRINTING
84	85	86	87	88	89	90	91	92	YEAR

Contents

How to use this book
How to use the programs
The story so far

EPISODE 1 1

The keyboard; SHIFT; RETURN;
CLR/HOME; LOADING a
program; LIST; PRINT;
INPUT; CREWDATA; LET; +; =;
memory location names

EPISODE 2 15

NEW; line numbers; RUN

EPISODE 3 25

DRINKSCODE; GOTO; \$,
strings/messages; IF... THEN

EPISODE 4 33

RND(); +, -, *, /; INT()

EPISODE 5 41

> <; number guessing game

EPISODE 6 49

FOR-NEXT; STEP; RIGHT\$;
LEFT\$; MID\$

EPISODE 7

59

OXYGEN; LEN(); ASC(); CHR\$()

Dr W Explains

73

Solutions

111

Index

115

How to use this book

This book is about an adventure — well, really two adventures! One adventure is in space. Our heroes must learn to control a spaceship's computer so that they can escape from an alien planet and get back home.

The second adventure has a far grander hero — you! In this adventure you will learn to control your own computer. It just so happens that the spaceship's computer is remarkably similar to your Commodore 64 and so you can learn along with Dr W, Aleat, and SN7. Fortunately one of the wanderers in space, Dr W, kept a notebook. When he got home our industrious friend wrote a whole series of explanations for his nephew Barnet who also has a Commodore 64. As you'll be quite interested in these explanations, we've copied most of them for you. You will find them in the section entitled "Dr W Explains." So whenever you find that you don't quite understand one of the computer-type words that are used, look it up in "Dr W Explains."

How to use the programs

In our story the three heroes find some computer programs. All the programs that our heroes use appear on the software in this package. The book will tell you when to LOAD each one.

The following are the programs that our heroes use (each one is also explained in the text):

CREWDATA

DRINKSCODE

OXYGEN

There are also four teaching programs that will help you with certain of the command words used in BASIC. You can use these programs when you like — at any stage of reading or even before you start the book. They will give you some extra help. Use them when you feel you need them.

- | | |
|----------|--|
| PRINT IT | Will help you to understand the PRINT command. |
| LINEOUT | Is about how line numbers work in BASIC. |
| STRINGY | This helps with the functions RIGHT\$(), LEFT\$(), MID\$(). |
| GUESSER | Here is a program that will teach you about the number guessing game that Aleat writes in Chapter 5. |

How to use the programs

In this story the three heroes find some computers programs. All the programs that are better than the others are in this package. The book will tell you when to use each one.

The following are the programs that are better than the others (they are better than the others).

CREWDATA

DRINKCOOL

DRINKCOOL

There are also some programs that will help you with some of the command words used in BASIC. You can use these programs when you like—at any stage of your programming. They will give you some extra help. Use them when you feel you need them.

PRINT IT We help you to understand the PRINT command.

LINEOUT is about how the numbers work in BASIC.

STRINGY The help with the function RIGHT\$(string, width).

CLASSER Here is a program that will look you about the names given to some of the things in BASIC.

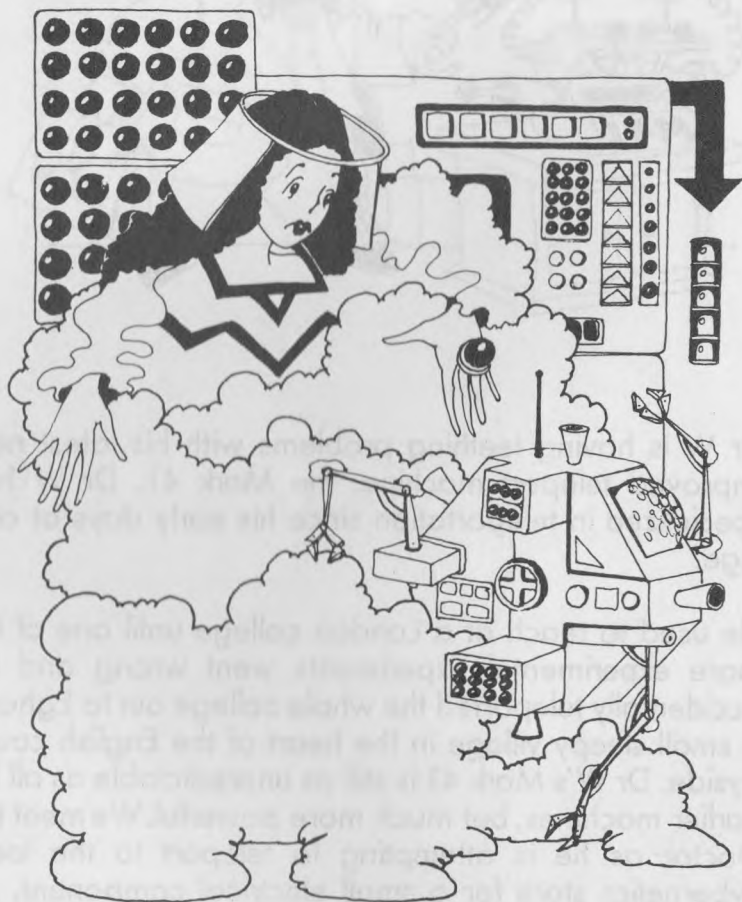
The story so far



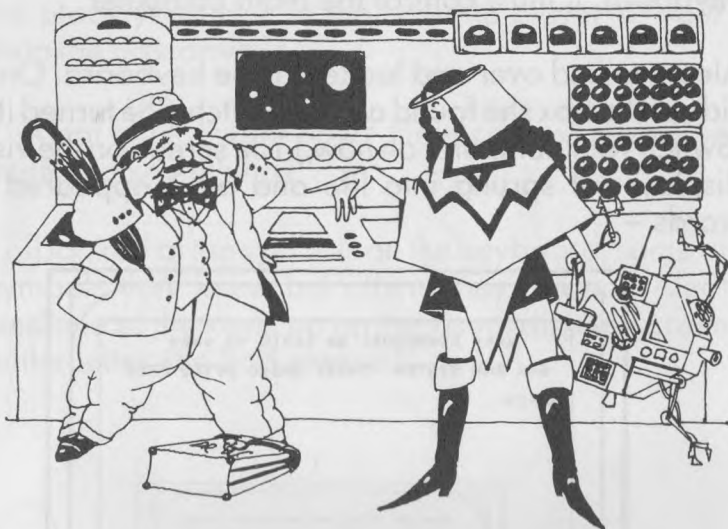
Dr W is having teething problems with his latest new improved teleport machine, the Mark 41. Dr W has specialized in teleportation since his early days at college.

He used to teach at a London college until one of his more experimental experiments went wrong and he accidentally teleported the whole college out to Egham, a small sleepy village in the heart of the English countryside. Dr W's Mark 41 is still as unpredictable as all his earlier machines, but much more powerful. We meet the Doctor as he is attempting to teleport to the local cybernetics store for a small electrical component. As usual his machine is not behaving itself. In order to solve the problem, Dr W gives his machine a short sharp blow with his umbrella. It whirrs into life and Dr W is picked up by the teleport beam.

Aleat, a young Arfonite (241 years old), and SN7, a small maintenance robot, are having problems with their spaceship's hyperdrive. Though they do not know it, their problems are caused by the amiable Dr W's teleport machine. Suddenly, and with a jolt, they are whisked away.



Episode 1



Our heroes materialize in the central control room of a deserted alien spaceship.

"Oh! Where am I?" wondered Aleat as she picked herself up from the dusty floor, "and who are you?"

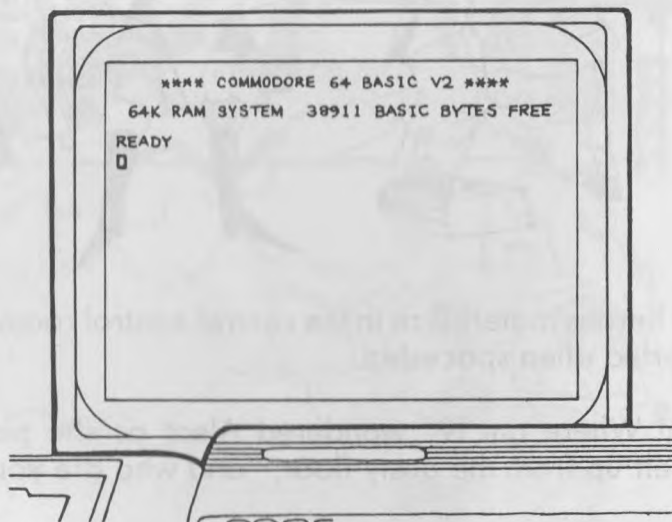
Dr W turned his head and rubbed his eyes. "Me? My name is Dr W. I'm most awfully sorry, I seem to have made some mistake. I was just teleporting to the cybernetics store and... and..." Dr W was at a loss for words. Meanwhile SN7 trundled across the floor.

"Get up, get on, there's work to do," it said over and over again.

Aleat pulled herself together, "It looks as though we'll have to work together to find out about this ship if we're ever going to get back home."

"Look," Dr W turned from a control panel that he had been looking at, "this seems to be some sort of keyboard, it must control the main computer."

Aleat walked over and looked at the keyboard. On the side of the box she found a small switch. She turned it on. (Switch your computer on now.) The screen on the visual display unit sprung into life and on it appeared the words –



Meanwhile Dr W had found a small black object on the floor in the corner. On the object was a torn label. "What's this?" he said.



"It is a form of storage media. It is obviously for this computation machine," said SN7 in its piercing voice. "The words on the label must be the names given to the stored programs and the device with the machine must be some form of reading device." SN7 snatched the object from Dr W and put it into the reading device. (You should place your tape in the cassette player or your disk into the disk drive.)

"So," Aleat said, "what now? How can we make the computer do something?"

They all looked at the symbols on the keyboard, some of the symbols they knew but others they did not. Aleat pressed one of the keys: up on the visual display screen came the letter she had pressed.





"Ah!" exclaimed Dr W. "The keys have more than one symbol on. Let's consider just the symbols on the tops of the keys. On some keys there is only one symbol on top but some keys have more than one symbol. Press one of those keys with two symbols on top."

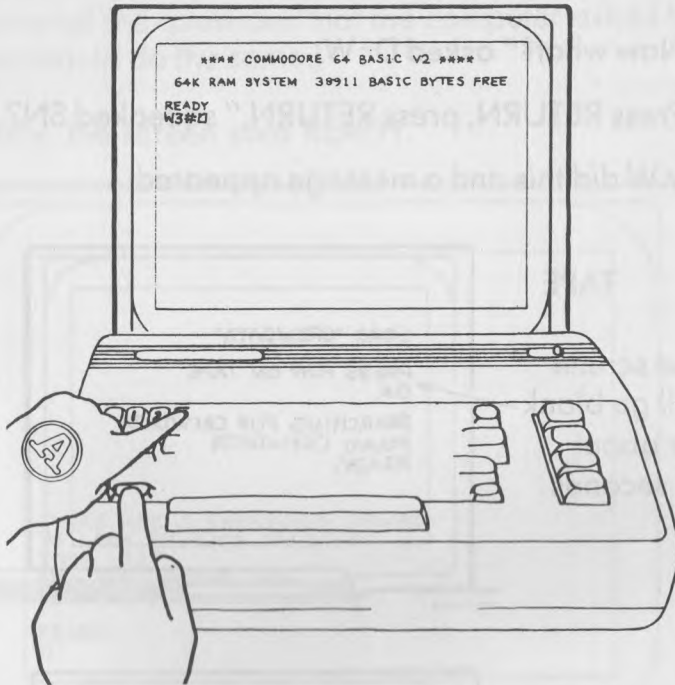
Aleat pressed the key with the number 3 on it and up on the screen appeared the lower symbol, the 3 shown on the key.



"How could we get the upper symbol?" asked Dr W.

SN7 piped up: "Hold down SHIFT, hold down SHIFT."

Dr W reached forward and held down the SHIFT key while Aleat pressed her key again. On the screen came the upper character.



"Now let's try to see what information is stored on the storage media," said Dr W.

"Analysis indicates that we must get the computer to load the information from the storage media."

SN7 interrupted: "CREWDATA! That was the first word on the label. And LOAD is right, LOAD is right!"

Dr W said, "SN7 is probably correct." He typed in

LOAD "CREWDATA"

If you have a disk type in

LOAD "CREWDATA", 8

"Now what?" asked Dr W.

"Press RETURN, press RETURN," squeaked SN7.

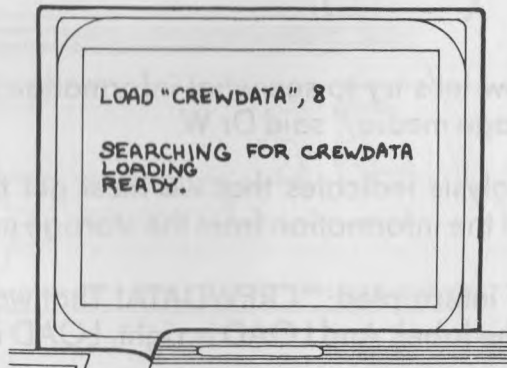
Dr W did this and a message appeared:

TAPE

The screen
will go blank
for about
30 seconds



DISK



Dr W looked puzzled as he said "what now?"
"RUN! RUN!" exclaimed SN7.

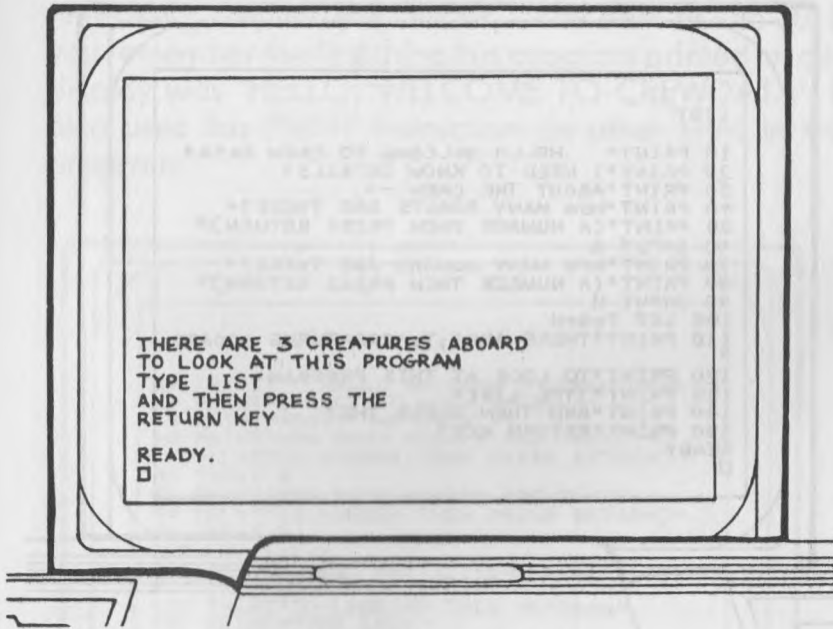
Aleat leaned over the computer and typed

RUN

"Press RETURN — don't forget," said SN7.

Then, on the screen appeared a question. They answered the questions that the computer asked them. (You should do the same.)

Finally, the screen said READY.



"What does that mean?" asked Dr W.

"Well," Aleat paused, "perhaps it means the computer is ready for something!"

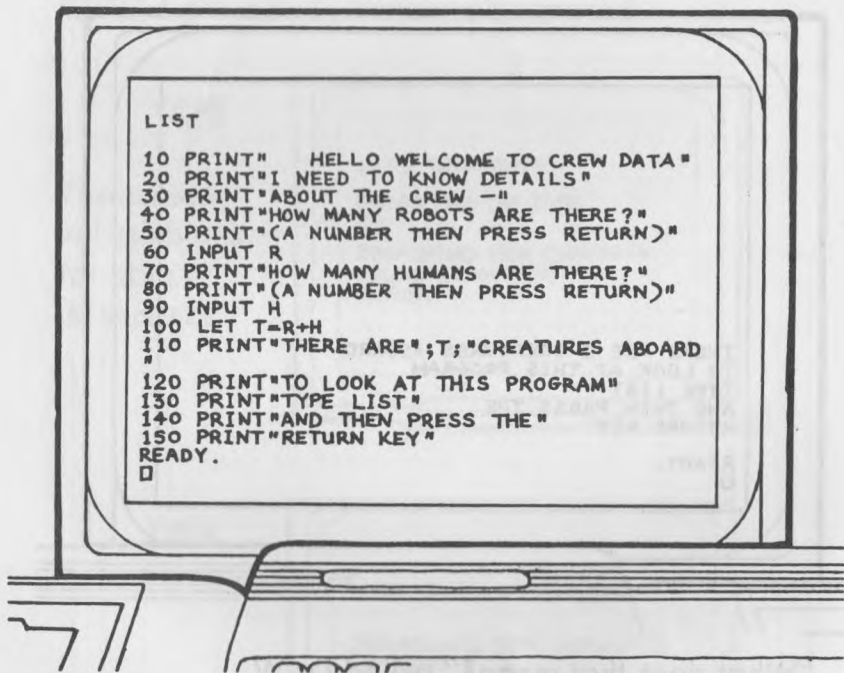
"It's ready for us, it's ready for us," interjected SN7.

"That's sensible." Dr W scratched his head. "But WHAT shall we do?"

Well it tells us to type LIST, that's what we should do." Aleat leaned forward and typed LIST. Nothing happened! (Have you typed LIST on your machine?)

"Why, it's done nothing!" exclaimed the Doctor. "Perhaps we should press the RETURN key." He reached over to the right of the keyboard and pressed RETURN.

Up on the screen appeared a list.



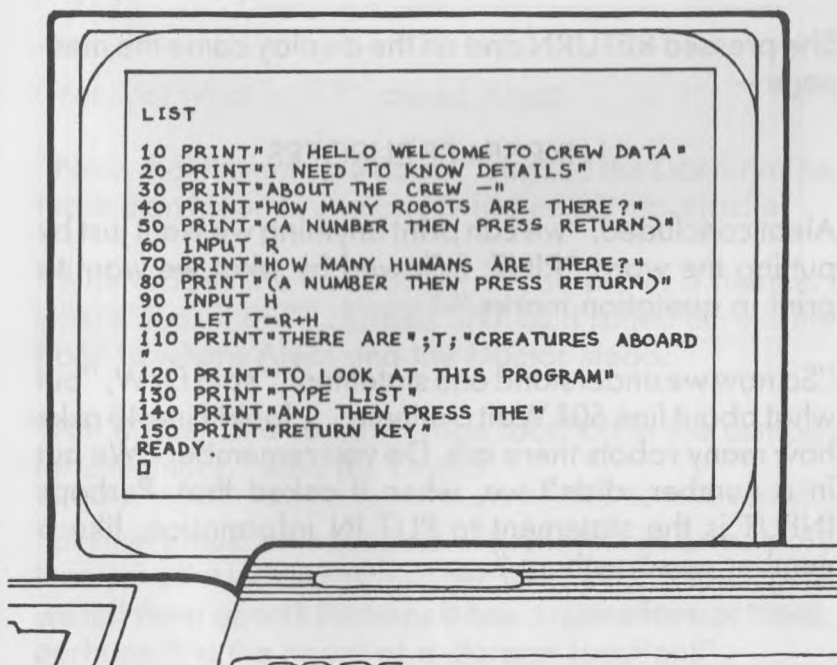
"Some sort of Sanscrit perhaps?" said the Doctor.

"I think not," replied Aleat, "this is obviously the program in question."

"Ah!" the Doctor's face lit up. "Now, each of these lines seems to start with a number. Also they seem to count up in tens, most curious." He pointed with his umbrella. "We may assume that each line is some sort of instruction: each line tells the computer to do one particular thing."

"O.K." Aleat took over. "So the computer should be able to understand one of these instructions if we type it in on its own."

"Precisely," replied the Doctor, "look here at line 10. If you remember the first thing this program printed on our display was 'HELLO, WELCOME TO CREWDATA'. It also uses this PRINT instruction on other lines in the program."



"So," said Aleat, "the computer should understand this PRINT command."

She stepped forward to the keyboard and typed in –

PRINT "HELLO"

Nothing happened! "Press RETURN," commented the Doctor. She did. Up on the screen came the word –

HELLO

"It worked!" the Doctor was jubilant.

"Wait, let's try something else." Aleat typed –

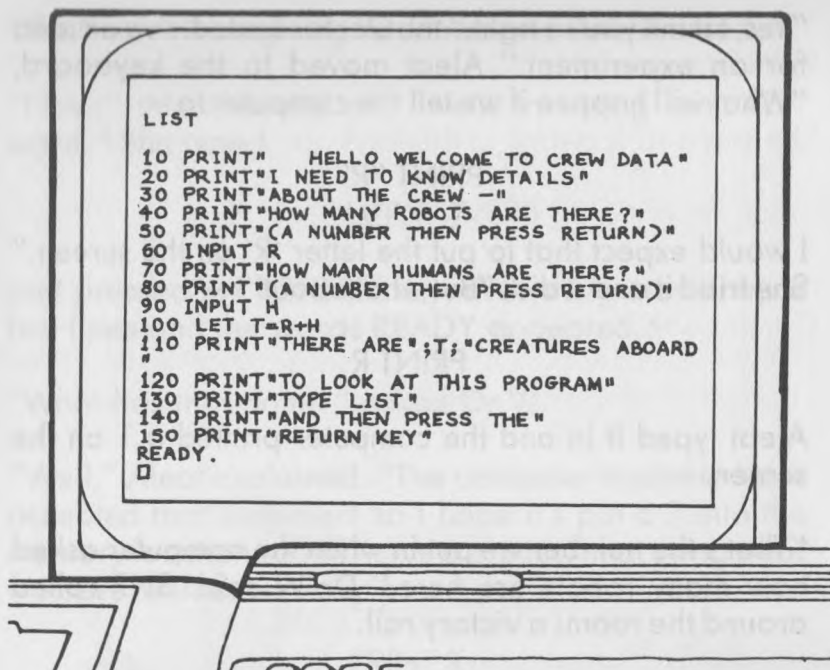
PRINT "I LIKE SPACE BURGERS"

She pressed RETURN and on the display came the message –

I LIKE SPACE BURGERS

Aleat concluded, "we can print anything we want just by putting the word PRINT, followed by what we want to print in quotation marks."

"So now we understand one statement," said Dr W, "but what about line 60? Wait a minute . . . look! Line 40 asks how many robots there are. Do you remember? We put in a number, didn't we, when it asked that. Perhaps INPUT is the statement to PUT IN information, like a number or something."



"Yes, but what is 'R'?" asked Aleat.

"Now, that I'm not sure about," replied the Doctor as he leaned on his umbrella and twiddled his moustache.

"Where does the number go, if you put in a number, where does it go?" chirped SN7 as it rolled across the floor to where Aleat and the Doctor stood.

"Ah! It goes into some storage location in the computer's memory," said the Doctor.

"But there must be thousands of such storage or memory locations," Aleat scratched her forehead. "How could we tell them apart? Perhaps it has a name for each one, perhaps R is the name of a storage location!"

"Yes, I think you're right," the Doctor smiled. I've an idea for an experiment." Aleat moved to the keyboard, "What will happen if we tell the computer to —

PRINT "R"

I would expect that to put the letter 'R' on the screen." She tried it and it did. "But what about" —

PRINT R

Aleat typed it in and the computer printed a 1 on the screen.

"That's the number we put in when the computer asked how many robots are here," Dr W said. SN7 rolled around the room: a victory roll.

"Ah yes," interrupted Aleat, "when we say 'PRINT R' we tell the computer to print the contents of memory location R."

"And when the R is inside quotation marks?" queried the Doctor.

"That's different," replied Aleat. "When the computer sees something inside those marks it prints it onto the screen exactly as it's written."

"Now line 100." Dr W squinted through his glasses. "This seems to be some form of mathematical expression: 'LET T=R+H' must mean add the numbers in R and H together and then put the answer in memory location T."

"Yes," said Aleat giving the Doctor a pat on the back.

"Now," Aleat continued, "we'll try using this 'LET' word." She typed

```
LET X=7
```

and pressed the RETURN key. Everything moved up a few lines and the words READY appeared.

"What has that done?" asked Dr W.

"Well," Aleat explained. "The computer seems to have accepted that statement so I hope it's put a 7 into the memory location called X. We can check that using what we have learned already if we type —

```
PRINT X
```

then 7 should come up."

"Yes, but it knows that X is 7 from the line above 'PRINT X' which says 'LET X=7'. What if we cleared the whole screen and then typed PRINT X?" asked the Doctor.

"Well, we'll do that." Aleat looked puzzled. "How can we clear the screen?"

SN7 rolled across the floor. "Hold down the SHIFT key then press the CLR/HOME key."

Dr W did as SN7 said and the screen cleared.

"Now try PRINT X," Dr W did as Aleat had said. He typed –

PRINT X

and, on pressing RETURN,

"It worked!" Aleat smiled.

"Time for food." Aleat pulled a couple of space burgers from her pocket and some moon cheese.



"Ah, lovely," Dr W could feel his mouth watering. Aleat and the Doctor sat down to eat their space burgers.

"Food, food, horrible, horrible!" SN7 screeched as it raced about the floor in frustration. The idea of a space burger in its circuits was too much for SN7!

EPISODE 2



"Now," said Dr W, "let's look at that program again." He typed LIST then pressed RETURN. As he had expected, the program came up on the screen.

If you have switched your computer off since Chapter 1, you will have to rewind the tape and wait for it to stop. Next hold down SHIFT while pressing CLR/HOME and then type LOAD "CREWDATA" (for disk users type LOAD "CREWDATA", 8). Press RETURN and do as the computer says. When it says READY, type LIST and press RETURN.



"What are these numbers at the beginning of each line?" Aleat asked. The Doctor was pondering the same problem.

His eyes lit up: "Ah! Each line seems to be an instruction to the computer. All the lines together form a list of instructions, hence the word LIST that we've used to make the computer display the list of instructions."

"We can probably put any list into the computer," said Aleat.

"Yes, but not all lists are lists of instructions," said the Doctor. "A shopping list isn't, it's a list of things. When it's a list of instructions that the computer can understand, we call that list a program."

"But how do the numbers fit in?" Aleat interrupted.

"Patience," Dr W went on. "Now, any list of instructions must always be done in a certain order. For instance, the instructions for eating a space burger:

REMARK THAT IT TASTES DELICIOUS
REMOVE WRAPPER

have a particular order. If you do them in the wrong order, you might not enjoy your meal! So, how do we tell the computer what order to do something in? We'll give each of the instructions a number, like 10 or 20. The computer does the instruction with the lowest number first and works its way through to the instruction with the highest number."

"So," said Aleat, "in the space burger example we might use —

500 REMARK THAT IT TASTES DELICIOUS
300 REMOVE WRAPPER

and the computer would do line 300 then line 500."

"Precisely," Dr W replied, "and we could easily add an extra instruction between the two we have here, like —

400 REMEMBER TO ADD SAUCE TO BURGER

The computer would then sort out the whole list into the correct order to form a nice tidy program.

300 REMOVE WRAPPER
400 REMEMBER TO ADD SAUCE TO BURGER
500 REMARK THAT IT TASTES DELICIOUS

Simple, is it not?"

"Well," Aleat looked puzzled. "Wouldn't it have been easier to just number the lines 1,2,3, and so on?"

"At first that would appear more logical," answered the Doctor, "but, if we had numbered our instructions 1, 2, 3, and so on, how would we fit in any extra instructions? In our example we numbered our first two instructions 300 and 500. This allowed us to put in a further instruction, namely instruction 400, later on —

400 REMEMBER TO ADD SAUCE TO BURGER

If we had numbered our first two instructions 1 and 2, we could not have put in the extra instruction!"

"So," Aleat said. "Since the program we were given on the tape is numbered in tens, we could put in extra lines just where we wanted. Let's try something." She typed —

15 PRINT "THIS IS RUBBISH"

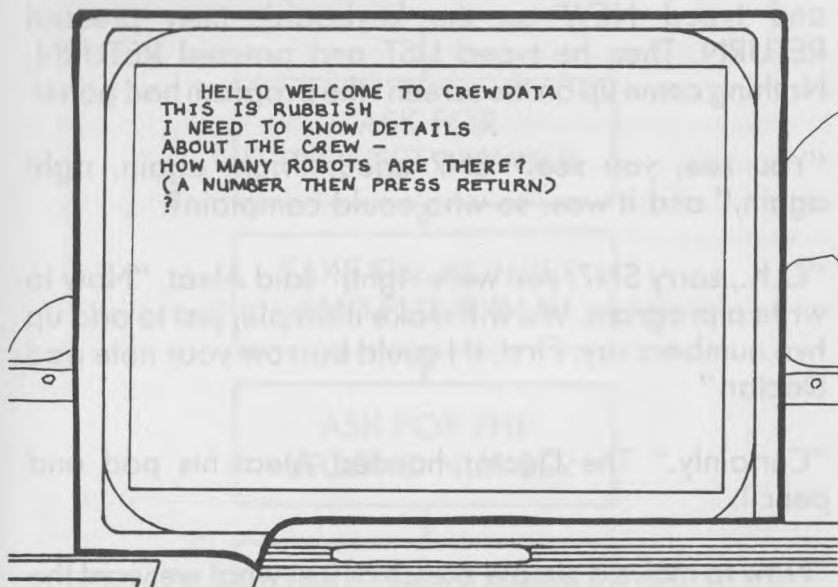
and pressed RETURN.

"Let's get the computer to run the program. I guess we could try typing RUN." She typed —

RUN

and pressed RETURN.

The screen sprang into life with –



"Well, very nice but it doesn't really make sense does it?" said Dr W.

"Dr W, I despair of your naivety. We have managed to modify a program on this ship's computer and all you can do is criticize."

"I'm most sorry," Dr W apologized. "Could we actually create a new program now?" he asked.

Aleat answered, "I see no reason why we can't, though we'll have to find out how to get rid of the old program before we can put our new program in. But how?"

"NEW, NEW, is the word," SN7 made another of its sporadic contributions.

"Shut up SN7." Aleat did not like loud interruptions. SN7 rolled into a corner. The Doctor walked forward and typed NEW on the keyboard, then pressed RETURN. Then he typed LIST and pressed RETURN. Nothing came up on the screen : the program had gone!

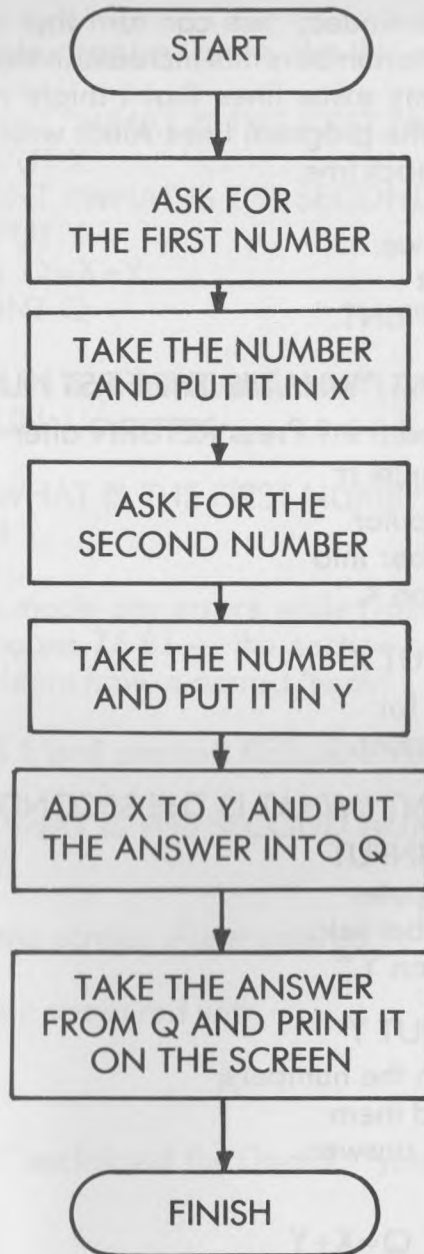
"You see, you see," SN7 cried, "right again, right again," and it was, so who could complain?

"O.K., sorry SN7, you were right," said Aleat. "Now to write a program. We will make it simple, just to add up two numbers say. First, if I could borrow your note pad Doctor."

"Certainly." The Doctor handed Aleat his pad and pencil.

"Now to make a simple sketch of just what we want the program to do." She started to draw.

Here is the sketch Aleat drew –



"Now," she continued, "we can turn that into a program. I'll use line numbers that increase in steps of ten so I can put in any extra lines that I might need." The following are the program lines Aleat wrote, with her comments for each line.

"To start with, we ask for the first number using PRINT."

10 PRINT "WHAT IS THE FIRST NUMBER"

(Have you typed it in? Press RETURN after each line.)

"Then we use INPUT to get the computer to put the number into memory location X."

20 INPUT X

"Next, we ask for the second number."

30 PRINT "WHAT IS THE SECOND NUMBER"

"Then we use INPUT to get the computer to put the number into memory location Y."

40 INPUT Y

"We have both the numbers so now we add them up and put the answer into Q."

50 LET Q=X+Y

"All that is left to do is PRINT the answer."

60 PRINT Q

"So the whole program looks like this —"

```
10 PRINT "WHAT IS THE FIRST NUMBER"  
20 INPUT X  
30 PRINT "WHAT IS THE SECOND NUMBER"  
40 INPUT Y  
50 LET Q=X+Y  
60 PRINT Q
```

She typed RUN and then pressed RETURN, to make the program RUN. Up came —

```
WHAT IS THE FIRST NUMBER  
?
```

(If you have made any errors while typing in this program, see pages 76-81 — the section entitled "EDITING" — to learn how to correct them.)

She entered 5 and pressed RETURN. Then —

```
WHAT IS THE SECOND NUMBER  
?
```

came onto the screen. Aleat entered 7.

The computer answered with —

12

"It worked!" exclaimed the Doctor, "you did it!" Aleat smiled.

The Doctor and Aleat experimented with the program for a little while. (Make sure you understand the program.)

EPISODE 3



"What now?" asked the Doctor.

"Shall we see what the other programs do?" enquired Aleat.

"Why not?" answered Dr. W.

If you have rewound your tape, then you should type —

LOAD "DRINKSCODE"

If you have a disk, type —

LOAD "DRINKSCODE" ,8

then press RETURN for both. Then when the computer says READY, type —

RUN

and press RETURN.

Then on the screen came the message —

IN ORDER TO USE THE DRINKS
MACHINE IN THE CORNER,
YOU MUST TELL ME
THE CODE WORD

"So that's what that bizarre piece of machinery is," commented Dr W.

"Yes, I could just do with an ice-cold drink." Aleat was thirsty. "But how can we find the code word?"

SN7 approached them. "Clue! Clue! Clue! It's five letters long — made from hydrogen and oxygen!"

"You know what it is, don't you?" Aleat cried, "tell us, we are thirsty."

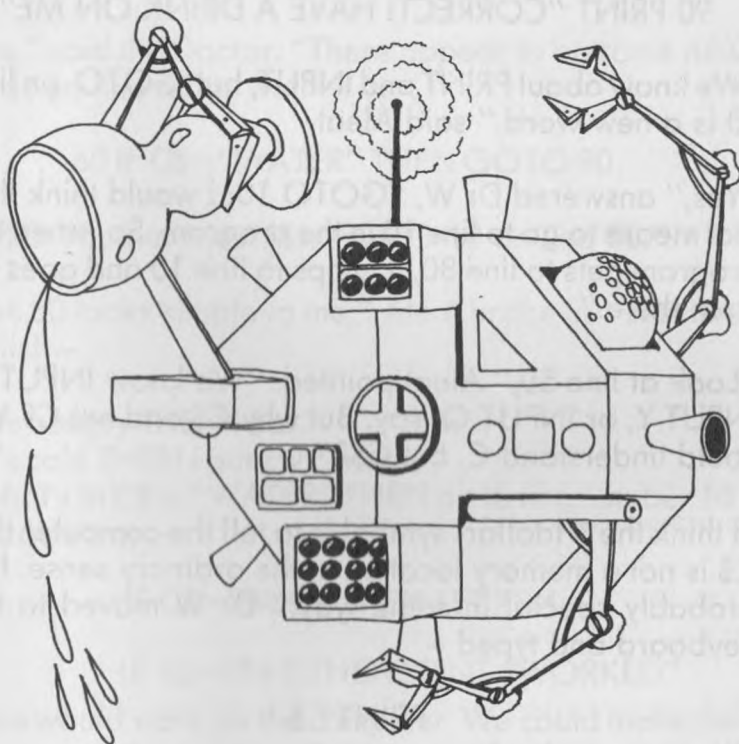
(See if you can guess what it is. Type in your guess and then press RETURN. If you get it right, the computer will tell you, otherwise it will put up the question again.)

"O.K.," answered the robot, "WATER!"

They typed in WATER and pressed RETURN.

CORRECT! HAVE A DRINK ON ME

the computer printed. They all had a drink except SN7 who rolled around saying "I hate liquid" very quietly so as not to be heard.



"I wonder how this program works?" The Doctor looked curious. "Let's type -

LIST

and see." Aleat typed LIST and pressed RETURN and up came the program:

```
10 PRINT "IN ORDER TO USE THE DRINKS"  
20 PRINT "MACHINE IN THE CORNER"  
30 PRINT "YOU MUST TELL ME"  
40 PRINT "THE CODE WORD";  
50 INPUT C$  
60 IF C$="WATER" THEN GOTO 90  
70 PRINT "WRONG"  
80 GOTO 10  
90 PRINT "CORRECT! HAVE A DRINK ON ME"
```

"We know about PRINT and INPUT, but GOTO on line 80 is a new word," said Aleat.

"Yes," answered Dr W, "GOTO 10; I would think that that means to go to line 10 in the program. So, when the program gets to line 80, it jumps to line 10 and goes on from there."

"Look at line 50," Aleat pointed. "We know INPUT X, INPUT Y, or INPUT Q, say. But why C\$ and not C? We could understand C, but C\$?"

"I think the \$ (dollar) symbol is to tell the computer that C\$ is not a memory location in the ordinary sense. It is probably special in some way." Dr W moved to the keyboard and typed –

PRINT C\$

The computer responded –

WATER

"That's the code word!" exclaimed the Doctor. "C\$ is special in that it stores words and not numbers." He leaned over the keyboard and typed –


```
LET Q$="AFTER DINNER"  
PRINT Q$
```

The computer printed –

AFTER DINNER

"Well, that seemed to work," commented Aleat.

"Yes," said the Doctor. "There appear to be some new words on line 60 as well –

```
60 IF C$="WATER" THEN GOTO 90
```

GOTO 90; that's just like GOTO 10 on line 80."

"Line 60 looks simple to me," Aleat broke in. "Just like English –

```
IF I'm hungry THEN I eat  
IF it's cold THEN I put my coat on  
IF what's in C$ is "WATER" THEN go to line number 10  
We could say –
```

```
IF C$="BUG" THEN LET X=4
```

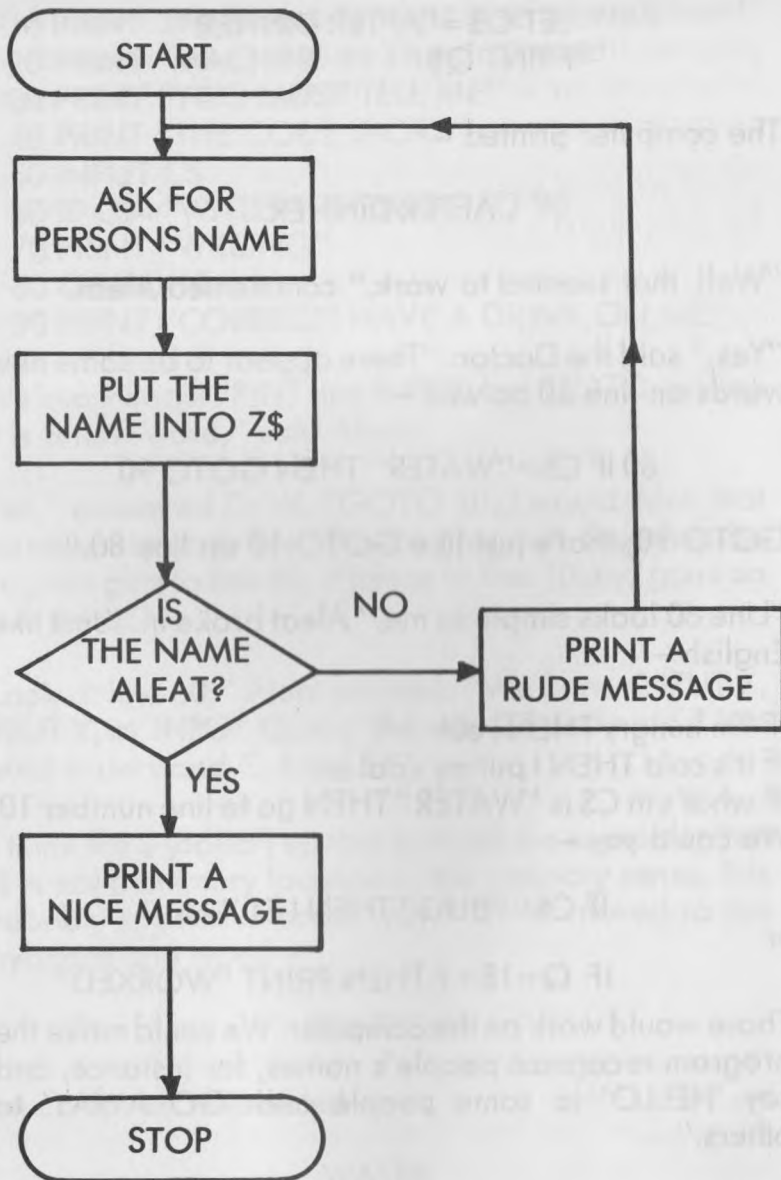
or

```
IF Q=18+Y THEN PRINT "WORKED"
```

Those would work on the computer. We could make the program recognize people's names, for instance, and say 'HELLO' to some people and 'GO AWAY' to others."

"Could you write such a program?" asked Dr W.

"Certainly." Aleat made a rough plan of how the program would work. Here it is.



"Now, here is the program," said Aleat. Here are the program lines with Aleat's explanations. Type each one in followed by RETURN. First type NEW to erase any program in memory.

"First we must ask for the person's name."

```
10 PRINT "WHAT IS YOUR NAME"
```

"Now we put the answer into a memory location called Z\$."

```
20 INPUT Z$
```

"Then we compare it with ALEAT, my name, and if it is me we go to line 70. I'm going to write that line later."

```
30 IF Z$="ALEAT" THEN GOTO 100
```

"We get to line 40 only if Z\$ wasn't 'ALEAT', so now we can print our rude message."

```
40 PRINT "GO AWAY"
```

```
50 PRINT Z$
```

"Now we go back to the start of the program."

```
60 GOTO 10
```

"If Z\$ was ALEAT
we jump to line 70
where we print a
polite message."

```
70 PRINT "HELLO"  
80 PRINT Z$
```

So, the whole program is –

```
10 PRINT "WHAT IS YOUR NAME"  
20 INPUT Z$  
30 IF Z$ = "ALEAT" THEN GOTO 70  
40 PRINT "GO AWAY"  
50 PRINT Z$  
60 GOTO 10  
70 PRINT "HELLO"  
80 PRINT Z$
```

"Now to run the program." Aleat typed –

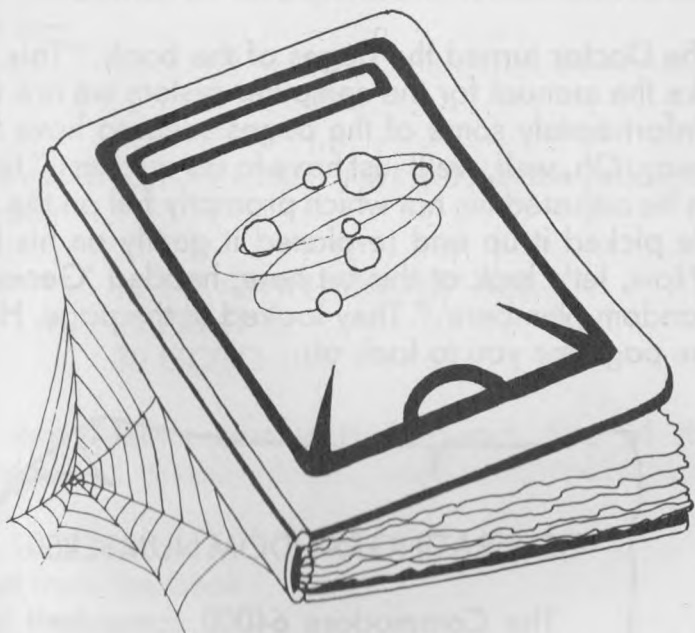
RUN

and pressed RETURN –

```
WHAT IS YOUR NAME  
?
```

the computer responded. (Experiment with this program; it will only recognize the name ALEAT. Try and make it recognize your name instead.)

EPISODE 4



"Look, look, look," SN7 spun in excitement, "a book, a book!"

They all came over to see what SN7 was looking at. It was the battered remains of a book. Dr W picked it up and blew the dust off. He opened the cover.

Inside, in handwriting that must have been 1000 years old, it said: "This book is the property of Captain Jane McCally."

"Fascinating, fascinating," murmured the Doctor. "Look — here is the publication date: 5782 A.D. This must be an old Earth ship! Well fancy that! I wonder why it was deserted?"

"Maybe it was the great plague of 229 S.S.C. (Standard Space Calendar)," said Aleat by way of explanation.

The Doctor turned the pages of the book. "This looks like the manual for the computer system we are using. Unfortunately some of the pages seem to have fallen away. Oh, well, we'll just have to do our best," he said as he adjusted his hat which promptly fell on the floor. He picked it up and re-placed it gently on his head. "Now, let's look at this bit here, headed 'Generating Random Numbers'." They looked at the page. Here is the page for you to look at:

GENERATING RANDOM NUMBERS

The Commodore 64000 computer has the capability of generating random numbers.

$$\text{LET } Q = \text{INT}(\text{RND}(0) * 100)$$

This expression is used in the traction mercury ion engine in order to generate numbers in the range of 0 to 99 to specify the random field ion injection point.

"Well, after reading that last bit, the expression –

```
LET Q=INT(RND(0)*100)
```

seems simple!" commended Dr W.

"Here's a chance for an experiment," Aleat said as she typed in –

NEW

and pressed RETURN, then she typed in the following program:

```
10 LET Q=INT(RND(0)*100)
20 PRINT Q
30 GOTO 10
```

Here are Aleat's comments for each line of the program:

"Line 10 is just copied from the book except that now it has a line number on it because it is part of my program."

```
10 LET Q=INT (RND(0)*100)
```

"This line prints out the value of Q so that we can see just what the instruction on line 10 has stored in memory location Q."

```
20 PRINT Q
```


"Then in line 30 we jump back to line 10 so as to do the process all over again."

```
30 GOTO 10
```

Aleat typed RUN (and pressed RETURN). Up on the screen came lots of numbers. As old ones disappeared off the top of the screen, new numbers appeared at the bottom. Aleat pressed RUN/STOP to stop the program.

"These numbers are all in the range 0 to 99," said Dr W.

"Yes," Aleat said. "The computer seems to make them up as it goes along."

"That's what is meant by generating random numbers," said Dr W. "It's just making up numbers. Oh, except they are between 0 and 99. We could experiment with that statement a bit: how about changing the 100 to a 50, that'll give us

```
LET Q=INT(RND(0)*50)
```

instead of –

```
LET Q=INT(RND(0)*100)
```

I would guess that would give a number in the range of 0 to 49."

"Let's try it." Aleat retyped line 10 to make it read:

```
10 LET Q=INT(RND(0)*50)
```

Then she ran the program.

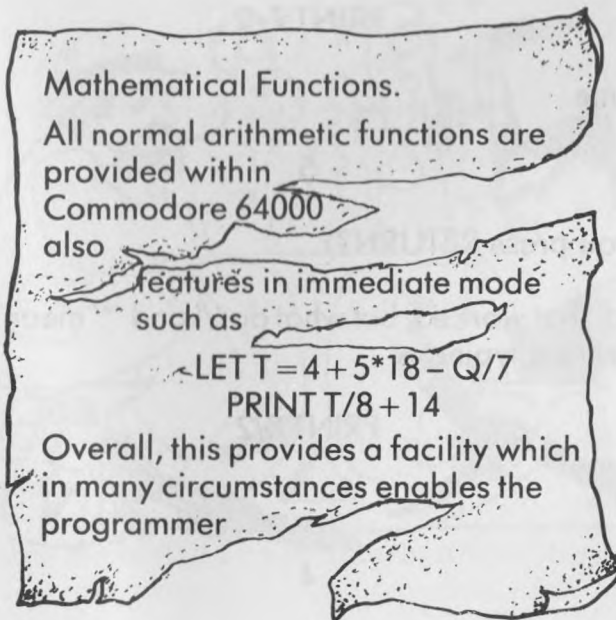
(You should know how to RUN a program by now, if you don't, here it is again. Type –

RUN

then press RETURN.)

Up popped more random numbers and, as Dr W had suspected, they were in the range 0 to 49. (Press RUN/STOP to see the numbers.)

"O.K. so that works, now where's the book?" asked the Doctor. Aleat handed it to him. "Thank you," said the Doctor. He turned the worn pages.



"Just what I was looking for!" exclaimed Dr W. "Here are all the math functions: how to add, subtract, multiply, and divide."

"Oh dear, there's a lot missing," commented Aleat, looking over the shoulder of the Doctor.

"Let's assume that

$$T=4+5*18-Q/7$$

and

$$T/8+14$$

are mathematical expressions. We already know what '+' means, and we may assume that '-' means minus." Just to check the Doctor typed -

PRINT 7-2

Up came

5

(Did you press RETURN?)

"Good, that worked, but what do '/' and '*' mean?" The Doctor tried typing -

PRINT 8/2

he got -

4

then

PRINT 10*5

he got

50

(Can you guess what '/' and '*' are?)

"It means that '/' is division and '*' is multiplication," Dr W made a few more notes. Then Aleat and Dr W sat down to a drink of triple thick star-shake, and relaxed for a while.



There is a difference between a man who is a man and a man who is a man. The difference is that a man who is a man is a man, while a man who is a man is a man.

The first difference is that a man who is a man is a man, while a man who is a man is a man.

(Can you guess what I am? I am a man.)
It means that I am a man, and I am a man. It means that I am a man, and I am a man. It means that I am a man, and I am a man.

The second difference is that a man who is a man is a man, while a man who is a man is a man.



EPISODE 5

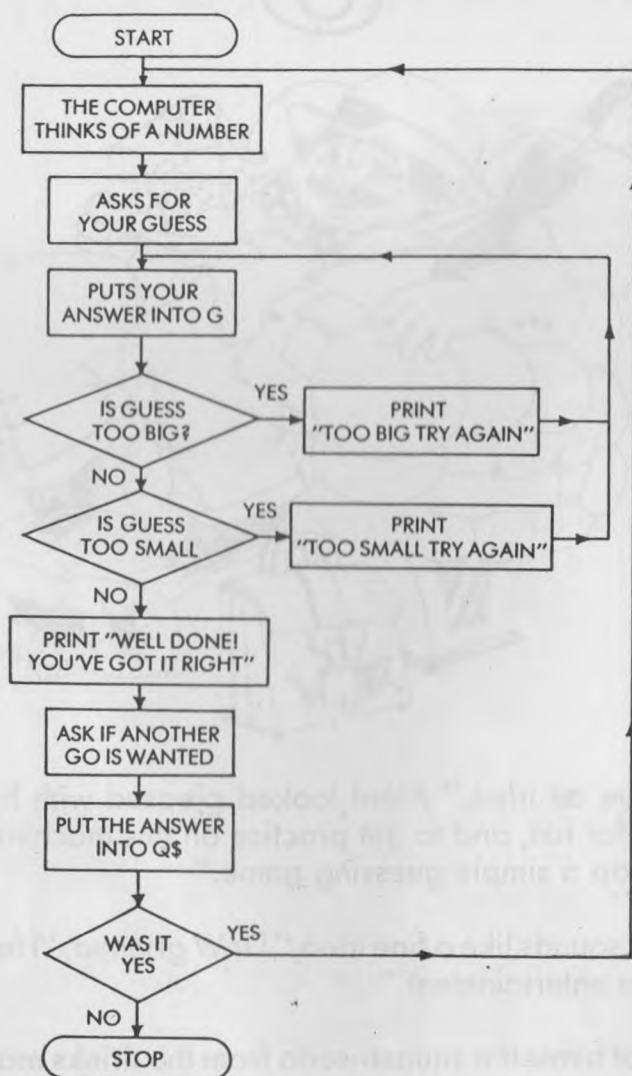


"I have an idea," Aleat looked pleased with herself. "Just for fun, and to get practice on this machine, let's develop a simple guessing game."

"That sounds like a fine idea," Dr W grinned, "I feel like a little entertainment."

He got himself a squash-soda from the drinks machine, with ice, and sank back into one of the super soft crew chairs.

"Now first let's draw a diagram to see how the program will look." She borrowed a sheet of paper from Dr W and drew. This is what she drew:



"Now," she continued, "we can turn this into a program."

Here is Aleat's program with her explanations:

First the computer makes up a number between 1 and 100 and puts it into memory location N"

```
10 LET N = INT(RND(0)*100)+1
```

"Now we ask the person playing the game to guess the number"

```
20 PRINT "CAN YOU GUESS THE NUMBER"
```

"The number that the persons puts in goes into a memory location called G"

```
30 INPUT G
```

"Next, we test G to see if it's too big and if it is we jump to line 110. I haven't written the line yet."

```
40 IF G>N THEN GOTO 200
```

"What's that symbol between G and N?" asked Dr W.

"On my home world, Arfon, we use that symbol to mean 'is bigger than,'" explained Aleat.

" $G > N$ means 'G is bigger than N' so –

40 IF $G > N$ THEN GOTO 110

In English, that means –

"If the number in memory location G is bigger than the number in memory location N, then go to line 110.

Aleat continued:

"We have tested to see if G is too big. Now we test to see whether G is too small. The symbol for 'is less than' is ' $<$ ' so the line is:"

50 IF $G < N$ THEN GOTO 130

"If our guess was too big we jumped from line 40 to line 110. If the number we guessed was too small we jump to line 130. So we are left with a G that is neither too big nor too small, so it must be just right. We must tell the user:"

60 PRINT "WELL DONE! YOU GOT IT RIGHT!"

"We could finish the game here, or we could ask whether the person wanted another game. I have chosen the second option:"

```
70 PRINT "DO YOU WANT ANOTHER GO"
```

"We take the answer, which will be YES or NO, and put it into a memory location called Q\$"

```
80 INPUT Q$
```

"Then we ask whether the answer was YES. If it was we go to the first line of the program, which is line 10, and start all over again."

```
90 IF Q$ = "YES" THEN GOTO 10
```

"If the answer wasn't YES then it was NO, so we stop"

```
100 STOP
```

"Now, back at line 40, we said that if the guess was too big we should jump to line 110, where we tell the player:"

```
110 PRINT "TOO BIG! TRY AGAIN"
```

"We go back to line 30 so that the player can put in another guess."

120 GOTO 30

"If the guess was too small we would jump from line 50 to line 130 and tell the player:"

130 PRINT "TOO SMALL! TRY AGAIN"

"Then we would go back to line 30 so that the player can put in another guess."

140 GOTO 30

"Here is the whole program."

```
10 LET N = INT(RND(0)*100) + 1
20 PRINT "CAN YOU GUESS THE NUMBER";
30 INPUT G
40 IF G > N THEN GOTO 110
50 IF G < N THEN GOTO 130
60 PRINT "WELL DONE! YOU GOT IT RIGHT!"
70 PRINT "DO YOU WANT ANOTHER GO";
80 INPUT Q$
90 IF Q$ = "YES" THEN GOTO 10
100 STOP
110 PRINT "TOO BIG! TRY AGAIN"
120 GOTO 30
130 PRINT "TOO SMALL! TRY AGAIN"
140 GOTO 30
```

"Right," said Dr W from his super soft crew chair. "Let's have a go." He leaned forward and typed RUN. Then he pressed RETURN.

"Now I'll bet you, Aleat," said Dr W, "that I can guess the number in less than 8 guesses every time."

How was the Doctor so certain of how many guesses it would take? Did he have a secret formula? As it happens, he did, and here is Dr W's explanation.

"Now," Dr W adjusted his hat. "If we had a guessing game that only told us if we got the right or the wrong answer then we would need to have 100 guesses to be certain of getting the right answer. Each time that we guessed wrong, we would know that the numbers we had already tried were wrong and nothing more.

In our game, it tells us if our number was too big or too small. Say we guessed 50 for our first guess, and it was wrong, and the computer told us TOO BIG . In that case we know 50 wasn't the number, and we know that 51 wasn't the number because that's bigger than 50 and 50 was too big. We also know that 52, 53, 54 and all the other numbers up to 100 are not the correct answer. That's a lot of information!

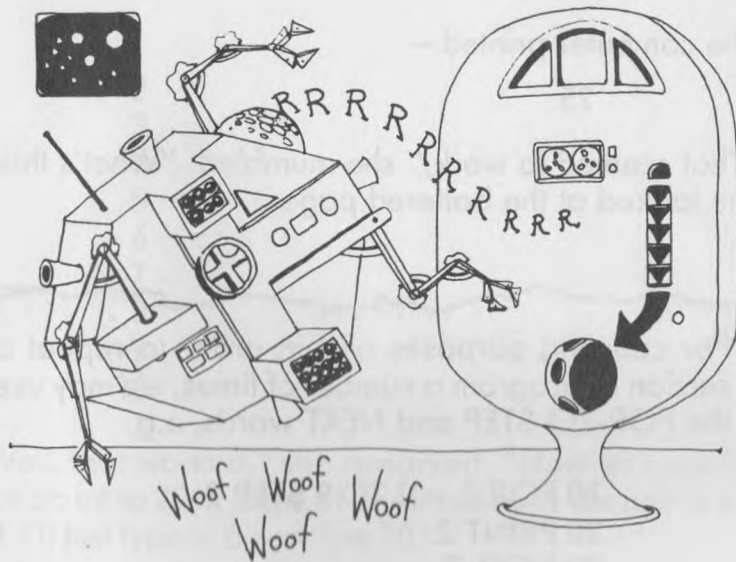
Now we are left with the numbers 1 to 49 to guess between. The best guess to make now is 25, which is about halfway between 1 and 49. Suppose that the computer tells us that 25 is TOO SMALL . We now know that 50 was too big, and that 25 was too small. Obviously, we must make a guess somewhere in between. The best place to guess is halfway between 25 and 50, which is about 38. We can keep on like this, guessing halfway between numbers which we know are wrong until we reach the right one!"

"That sounds terribly complicated!" said Aleat. "I think I'll just guess what I feel like guessing."

The Doctor won most of the games.



EPISODE 6



Dr W is keeping himself amused in his super soft crew chair with Aleat's number game. SN7 is trying to talk to the drinks machine in the same way a dog might bark at a stranger. And Aleat is leafing through Captain Jane's manual. She is mumbling to herself. "Now, if I wanted to print all the numbers from 1 to 10, I could use a program; for instance the one shown here:

```
10 LET C = 0
20 LET C = C + 1
30 PRINT C
40 IF C < 10 THEN GOTO 20
```


Line 20 is interesting: it must mean take the number in (memory location) C and add 1 to it, then put the answer back into the same place (memory location C). I think I'll check that that works." She typed (without line numbers)

```
LET C = 24
LET C = C + 1
PRINT C
```

The computer printed –

25

"That seemed to work," she mumbled. "What's this?" She looked at the battered page:

For counting purposes and in order to repeat a section of program a number of times, we may use the FOR-TO-STEP and NEXT words, e.g.

```
10 FOR Z = 1 TO 9 STEP 2
20 PRINT Z
30 NEXT Z
40 PRINT "THAT'S ALL FOLKS!"
```

This short program will print –

```
1
3
5
7
9
THAT'S ALL FOLKS!
```

"So," Aleat said to herself, "instead of counting to 10 using LET, IF-THEN, and GOTO, we could use FOR, STEP, and NEXT. She typed in NEW and then:

```
10 FOR Z = 1 TO 10 STEP 1
20 PRINT Z
30 NEXT Z
```

This would print —

```
1
2
3
4
5
6
7
8
9
10
```

"Well, that worked," she remarked. "Now let's see if it can count to 20 in steps of 2: All I need to change is line 10. I'll just type in a new line 10."

"Yes," added the Doctor, "if you type in a new line with the same number as an old one, the new line will replace it."

Aleat typed in:

```
10 FOR Z = 0 TO 20
```

and then LIST, which gave her:

```
10 FOR Z = 0 TO 20
20 PRINT Z
30 NEXT Z
```

She entered RUN and pressed RETURN. That printed —

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20
```

"Oh dear! I wanted it to count in two's. What have I done wrong?" said Aleat. "Oh — silly me! I've left off the STEP command. Wait a minute!

"Even though I hadn't told it what STEP size to use, it counted in ones! It must suppose you want a STEP size of one if you don't tell it. That's clever! Anyway," she continued, "to get back to counting in two's, I'll type in a correct line 10 —"

```
10 FOR Z = 0 TO 20 STEP 2
```

This time it worked properly.

"We could use FOR ... NEXT to print something 100 times –

```
10 FOR Z = 1 TO 100
20 PRINT "ROUND AND ROUND"
30 NEXT Z
```

That would print –

ROUND AND ROUND

one hundred times. This statement could be useful for counting with." She went on looking through the manual:

It is often useful to chop up words. There are three statements for doing this –

```
1 RIGHT$
2 LEFT$
3 MID$
```

First we will look at RIGHT\$

"Oh dear, a bit is missing!" she groaned.

```
10 LET A$ = "BATTLE"
20 LET B$ = RIGHT$ (A$,2)
30 PRINT B$
```

This would produce

The next page was missing as well so Aleat had to resort to experiment. She ran the above program. (You should, too.)

"Hmm," she said. "LE. It has taken the two rightmost letters of BATTLE and put them into B\$," She then changed line 20 to –

```
20 LET B$ = LEFT$(A$,2)
```

On running the program again, she got –

```
BA
```

"So, if I replace that 2 by a 3, I would get BAT." She did it.

```
20 LET B$ = LEFT$(A$,3)
```

And sure enough, on running the program again, she got –

```
BAT
```

"So LEFT\$(A\$,3) takes the 3 leftmost characters of A\$ in this case. To take the 5 leftmost characters of say, Q\$, we would use LEFT\$(Q\$,5). Hmm . . ."

Aleat turned back to the book. Not much of the writing about the MID\$ statement was there, but here is what Aleat saw:

```
140 LET Q$ = LEFT$(C$,14)
150 LET T$ = MID$(C$,4,3)
160 LET F$ = Q$+T$
170 LET S$ = ""
180 PRINT "STAND BY FOR"
```

She experimented by writing a short program to test MID\$ –

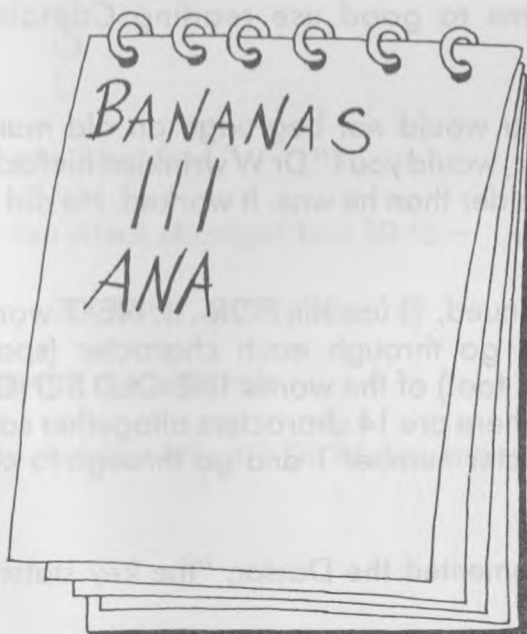
```
10 INPUT A$  
20 LET B$ = MID$(A$,2,3)  
30 PRINT B$
```

Running this, up came a ?. She put in the word BANANAS to which the computer replied –

ANA

Dr W had stopped playing the guessing game some time ago and looked on with interest.

“Ah!” Dr W pulled out his note book and drew



"So?" Aleat look puzzled.

"Well," said the Doctor, "ANA is in the word BANANAS, isn't it? If we start on the 2nd letter of BANANAS and then take 3 letters we get ANA, see? That's what MID\$(A\$,2,3) means."

"I see," Aleat carried on, "so if TIME was stored in A\$, the second letter is I and taking three letters from there would get IME. That's useful as we could look at one letter of a word at a time." Aleat wrote a program to test her theory.

```
10 LET Q$ = "THE OLD SCHOOL"  
20 FOR L = 1 TO 14 STEP 1  
30 PRINT MID$(Q$,L,1)  
40 NEXT
```

"You see Dr W, while you were playing number games I put my time to good use reading Captain Jane's manual."

"Surely you would not begrudge an old man a little amusement, would you?" Dr W wrinkled his face to look ten years older than he was. It worked. He did look ten years older.

Aleat continued, "I use the FOR . . . NEXT words that I learned to go through each character (spaces are characters, too!) of the words THE OLD SCHOOL one at a time. There are 14 characters altogether so we start with character number 1 and go through to character 14."

"Yes," commented the Doctor, "the key statement is:

MID\$(Q\$,L,1)

That just takes the words in Q\$, THE OLD SCHOOL, and starting at the 1st, 2nd, 3rd... 14th, (whichever L tells it to) it takes one letter so, when L is 3 we get E, when L is 4 we get a space, and when L is 5 we get O." Aleat ran the program and up came —

T
H
E

O
L
D

S
C
H
O
O
L

"There, it worked."

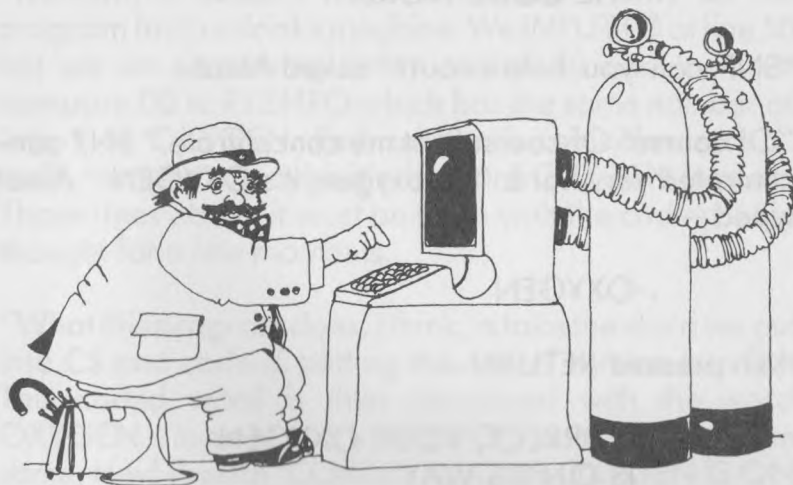
Just for fun Aleat changed line 30 to —

30 PRINT MID\$(Q\$,L,14)

and then ran the program.

"What a peculiar effect," Dr W was amused.

EPISODE 7



Dr W wanted to use the next program.

If you have the tape, then you should type —

LOAD "OXYGEN"

If you have the disk, type —

LOAD "OXYGEN", 8

then press RETURN. Then when the computer says
READY, type —

RUN

and press RETURN.

Then, the following message came up on the screen —

IN ORDER TO TURN ON
THE OXYGEN SUPPLY
YOU MUST TELL ME
THE CODE WORD

"SN7, can you help us out?" asked Aleat.

"Of course! Of course! Let me concentrate." SN7 concentrated very hard. "It's oxygen, it's OXYGEN." Aleat typed —

OXYGEN

then pressed RETURN —

CORRECT, YOUR OXYGEN
IS ON ITS WAY

READY

Aleat LISTed the program.

```
10 PRINT "IN ORDER TO TURN ON"  
20 PRINT "THE OXYGEN SUPPLY"  
30 PRINT "YOU MUST TELL ME"  
40 PRINT "THE CODE WORD"  
50 INPUT C$  
60 LET D$ = ""  
70 FOR Z = 1 TO LEN(C$)  
80 LET D$ = D$ + CHR$(ASC(MID$(C$,Z,1)) + 1)  
90 NEXT Z  
100 IF D$ = "PYZHFO" THEN GOTO 130
```

```
110 PRINT "WRONG"  
120 GOTO 40  
130 PRINT "CORRECT, YOUR OXYGEN"  
140 PRINT "IS ON ITS WAY"
```

"So, that's the program." Dr W looked carefully. "Mmmm, it doesn't work in the same way as the program for the drinks machine. We INPUT C\$ at line 50 but we do something funny at line 80 and then we compare D\$ to PYZHFO which has the same number of letters as OXYGEN. Perhaps that's coincidence or ... wait... maybe... maybe it's the word OXYGEN in code. Those lines above it must be to do with the coding." He thought for a few moments.

"What this program does, I think, is take the word we put into C\$ and code it, putting the coded version into D\$. This coded word is then compared with the word OXYGEN – in code that's PYZHFO – and if they are the same, Hey! Presto! 'CORRECT, YOUR OXYGEN IS ON ITS WAY' says the computer," Dr W finished.

"Lines 60 to 90 are the coding part of the program I think," said Aleat –

```
60 LET D$ = ""  
70 FOR Z = 1 TO LEN(C$)  
80 LET D$ = D$ + CHR$(ASC(MID$(C$,Z,1))+1)  
90 NEXT
```

"Line 60 –

```
60 LET D$ = ""
```

We've seen things like –

```
LET D$ = "FRED"
```

but not –

```
LET D$ = ""
```

I think what this does is to put no message at all into D\$. It sort of cleans it out or empties it."

Aleat continued, "Now what does LEN (C\$) mean on line 70? Let's try typing —

```
PRINT C$
```

that prints

```
OXYGEN
```

Now –

```
PRINT LEN(C$)
```

That prints

```
6
```

there are 6 letters in OXYGEN. Let's try –

```
PRINT LEN("HELLO")
```

that prints –

```
5
```

So LEN () gives us the total LENgth of a message. The message to check can be stored in a memory location, or just written in quotes as we found with HELLO.

Now, the whole of line 70 –

```
70 FOR Z = 1 TO LEN(C$)
```

is the start of a loop. The loop goes round once for each letter of the word in C\$.

If C\$ was SKID say, then LEN (C\$) would be 4. The loop would go round setting Z first to 1, then 2, then 3, and then 4."

"O.K." the Doctor interrupted, "What does line 80 do?" They looked at line 80 –

```
80 LET D$ = D$+CHR$(ASC(MID$(C$,Z,1))+1)
```

and gulped. It's a long line!

"Oh dear!" commented Aleat.

"Well, we'll start at the beginning." Dr W adjusted his hat, which promptly fell on to the floor. He continued undaunted. "First –

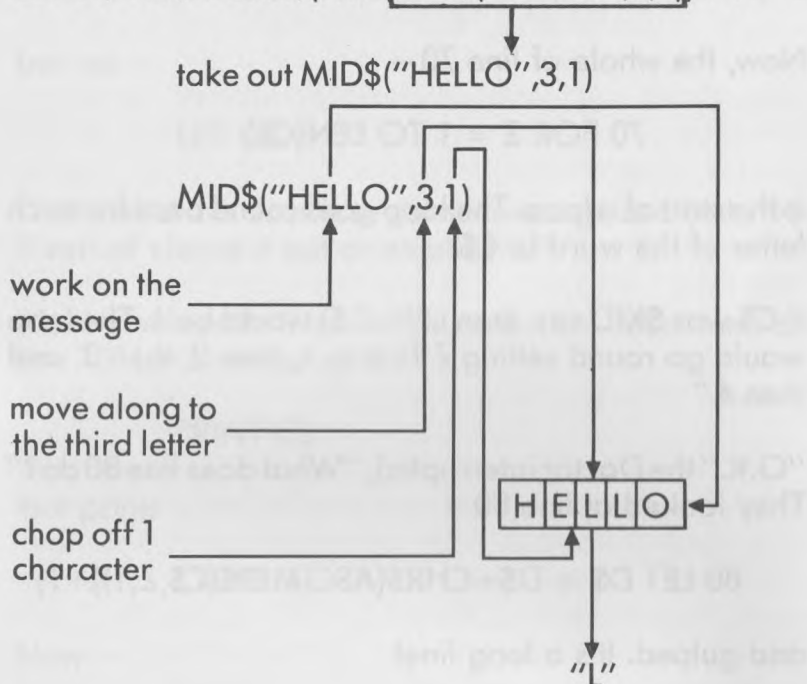
```
CHR$(ASC(MID$(C$,Z,1))+1)
```

Let's work it out in stages. First we'll replace C\$ with HELLO, and then replace Z with 3. That gives —

```
CHR$(ASC(MID$("HELLO",3,1))+1)
```


"Now to take that to pieces!

```
LET D$ = D$ + CHR$(ASC(MID$("HELLO",3,1)) + 1
```



"I see," interrupted Aleat, "so

```
MID$("HELLO",3,1) = L
```

Let's replace all that with 'L' so that line 80

```
LET D$ = D$ + CHR$(ASC(MID$("HELLO",3,1)) + 1)
```

changes into:

```
LET D$ = D$ + CHR$(ASC( "L" ) + 1)
```

or:

```
LET D$ = D$ + CHR$(ASC("L") + 1)
```

"Now let's examine ASC," said the Doctor. "Suppose we try:

```
PRINT ASC("B")
```

That printed a 66 on the screen. Now let's try:

```
PRINT ASC("L")
```

He typed it in and it printed the following:

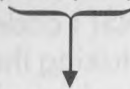
76

"Mmmm," the Doctor was intrigued.

"Hey!" Aleat exclaimed, "A.S.C. stands for Associated Space Code! Each letter has a special code number to represent it. All the ASC does is give us the code number of the letter in its brackets, so when Z was 3 and C\$ was HELLO we had the code number for L which is 76.

"Let's draw that out again," suggested Aleat. "Take the latest version of line 80 and replace ASC ("L") with 76."

```
LET D$ = D$ + CHR$(ASC("L") + 1)
```



```
LET D$ = D$ + CHR$(76 + 1)
```

"Good," said the Doctor, "that brings us to 'CHR\$' for the next puzzle. Let's try another experiment." He typed in:

```
PRINT CHR$(76)
```

and the letter "L" appeared on the screen.

"So," said Aleat, "CHR\$ seems to do the reverse of ASC. In CHR\$ we put in a number and we get out a character. With ASC we put in a character and get out a number. The command CHR\$ must be short for character then. In both cases the number is the code for the letter. Now look at our line again —

```
LET D$ = D$ + CHR$(ASC(MID$(C$,Z,1)) + 1)
```

They sat and looked at the screen thoughtfully as SN7 quietly rolled around and around making a gentle rattling noise.

Suddenly the Doctor arose, startling Aleat and making SN7 stop dead in its tracks. "I think ...," he mumbled thoughtfully, "I think I can see something!"

Aleat looked up at the Doctor with hope as he added, "It's not CHR\$(something) that the program uses, it's CHR\$(something + 1) — that's it, that's it! Look, if CHR\$(76) = "L" then we have to find CHR\$(76 + 1) or CHR\$(77) — that's 'M'."

"I see too!" exclaimed Aleat. "The program is adding one to the CHR\$() code each time, which really means that it's simply taking the next letter." She moved over to the keyboard and typed in:

```
PRINT CHR$(76)
```

and an "L" appeared on the screen. She then retyped it to read:

```
PRINT CHR$(76 + 1)
```

and on pressing RETURN, the letter "M" appeared. "Aha!" they exclaimed in unison as SN7 trundled up proclaiming squeakily, "Type it in, type it in, type it in!"

"Yes Aleat, all we need to do is move each letter of PYZHFO back one in the alphabet to get OXYGEN." The Doctor took out his fountain pen and wrote in his notebook:

PYZHFO

Then mumbling away as he did so, he calculated: "The letter before P is O, that before Y is X, the one before Z is Y, before H is G and before..."

"That's it, that's it!" cried Aleat jubilantly, "that's how the word OXYGEN is made, it's obvious!" "Look," she said scribbling away on Dr W's notebook:

P	Y	Z	H	F	O
↓	↓	↓	↓	↓	↓
O	X	Y	G	E	N

In spite of their triumph, Aleat still had a puzzled look on her face.

"There's still one thing missing," she murmured. "That last line we looked at, line 80, we've still not fully solved it. What about the first part:"

LET D\$ = D\$ + etc.

I'm not sure about that."

Dr. W took over. "I think it's probably similar to the number versions of the same thing:

```
LET X = X + 1
```

Let's do an experiment -

```
LET D$ = " "
```

That's to clean out D\$ so that it contains no letters.
Now -

```
LET D$ = D$ + "S"
```

```
PRINT D$
```

We get -

S

And if I type -

```
LET D$ = D$ + "N"
```

```
PRINT D$
```

I get -

SN

Then

```
LET D$ = D$ + "7"
```

```
PRINT D$
```

which gives

SN7

So, you see, each time we add an extra letter. With —

```
LET D$ = D$ + CHR$(ASC(MID$(C$,Z,1))+1)
```

We add to D\$ the letters P, Y, Z, H, F, and O, one at a time each time we go round the FOR-NEXT loop. Hence we end up with the word PYZHFO in D\$, given that OXYGEN was the word in C\$ to start with. "Now," continued the Doctor, "on line 100 we see if D\$ is PYZHFO and if it is, we go to line 130. The rest of the program is easy, as it is the same as the drink machine program."

What with fresh oxygen and a good supply of drinks all seemed well. But still they had not figured how to set the ship into motion. Dr W looked worried. "How can we get this ship moving?" he said.

"Can't a man of your calibre, who can build teleport machines, rewire a spaceship," asked Aleat? "Well — we can now use the computer to navigate, and trajectory calculations are no problem. They'll be the same as on my own ship."

"I'll do the programming, you worry about starting the engine," said Aleat. But Dr W had already set to work. His lower half could be seen sticking out of a service hatch. This picture of greatness was accompanied by muffled words:



"This green one goes to the CPU, and the blue switch controls the burst modulator."

Dr W was busy for hours. Aleat meanwhile had finished her programming and had settled down and gone to sleep.

Until...

"Goodness gracious! Its a dynoburst reversed polarity anti-quark annihilation drive," exclaimed the Dr. "Most impressive!" He crawled out of the inspection hatch and stood up. Reaching forward he pressed a few buttons on the control console.

"Prepare for acceleration," came a synthesized voice. Dr W sat in one of the chairs and Aleat woke up. Suddenly there was a rumble in the distance.

"10... 9... 8..." came the synthesised voice, "7... 6... 5... 4... 3" The rumble turned into a roar -

"2"

the roar turned into a scream -

"1"

The scream turned into a sound like a thousand hurricanes -

"0"

Dr. W and Aleat were pressed hard back into their chairs and the ship was already far, far away from where it had been.

Zegonenthon looked at his radar and cursed as his standard type five missiles exploded in the spot where Aleat and the Doctor once were.

DR W. EXPLAINS

... being a logical, concise, and rational explanation of the art of programming the computational machine referred to in common parlance as the Commodore Sixty-Four.

To Barnet, a distillation of the wisdom and intelligence gleaned by your dear Uncle during my enforced sojourn in an alien land.

Within these manifold pages you will find my humble gleanings upon the following esoteric topics:

ASC()	NEW
CHR\$()	PRINT
COLON(:)	REM
EDITING	RND()
END	SAVE
FOR. . .NEXT	SEMICOLON(;
GOTO	STEP
INPUT	STOP
INT ()	STRING
LEN()	VARIABLE (see LET)
LET and MEMORY LOCATION	VERIFY
LIST	solutions to exercises
LOAD	
MATHEMATICAL SYMBOLS	
MID\$()	
MULTI-STATEMENT LINES	

ASC()

Your computer has a special code which it uses in its electronics. Every letter, number, and symbol (like *) on the keyboard has its own code number which the computer uses whenever it does anything with any of them. The command ASC() gets the computer to tell you its code number for one of them.

For example, typing –

```
PRINT ASC("A")
```

and pressing RETURN will give you 65. That is the computer's code number for the letter A .

The character you are using must always be between quotes ("), if not the computer won't understand.

Try typing –

```
PRINT ASC("1")
```

and pressing RETURN. You should get your computer's code number for '1' which is 49. You can also do things like this:

```
10 LET B = ASC("7")  
20 PRINT B
```

which when RUN will put the code number 55 onto the display, 55 being the code number for the number 7.

The command ASC() is used by the coding program in Chapter 7. The opposite to ASC() is CHR\$() which

changes a code number into its actual symbol (see `CHR$()`).

Incidentally, ASC is short for ASCII, which in turn is short for American Standard Code for Information Interchange. (This is a term that was used quite a bit back in the twentieth century.)

CHR\$()

Your computer has a special code which it uses in its electronics. Every letter, number, and symbol (like *) on the keyboard has its own code number which the computer uses whenever it does anything with any of them. The command `CHR$()` gets the computer to tell you the character that corresponds to a given code number.

For example,

```
PRINT CHR$(66)
```

will put a letter B onto the screen. In your computer's own special code, the number 66 is the code word for the letter B. All the symbols on the computer's keyboard have their own code number. For instance, can you see what this does? —

```
PRINT CHR$(13)
```

Well, 13 is the code number for the RETURN key, so, as your computer can't write RETURN without being told to `PRINT RETURN`, it has actually done a RETURN instead, which makes the computer start a new line: the word READY is one line further down the screen than it usually is after a `PRINT` command.

You can also do things like

```
LET X$ = CHR$(67)
```

which will store the letter C in memory location X\$.

The opposite to CHR\$() is ASC() which turns a symbol into its code number (see ASC()). Both CHR\$() and ASC() are used in the code program of Chapter 7.

COLON

A sign ":" used to separate items. (See Multi-Statement lines on page 98).

EDITING

What do you do when you've made a mistake and want to change it? You can type in the whole line again, but that can be tiresome. Well, your Commodore 64 can help you here. It has what are called cursor control keys at the bottom right of your keyboard. These are shown below.



The cursor is the flashing square on the screen that tells you where you will be writing next. The cursor control keys, naturally enough, control the cursor! With them you can move the cursor anywhere you like in the writing area of the screen. Try pressing the keys a few times now.

Well, that's all very well — but what when you've got to the bottom of the screen? Try pressing the keys holding down SHIFT. Now the keys move the cursor in the opposite direction — hence the two arrows on each key. Press. RETURN when you've finished, so the computer knows.

So – how do you correct a mistake? Well, it depends partly upon what's wrong. Suppose you've gone and typed INPUT C\$ when you meant PRINT C\$ (type this in, EXACTLY as written here, and press RETURN when you've finished the line):

```
10 INPUT C$
```

Now to change the INPUT to a PRINT, we do the following:

- hold down SHIFT and press the up-arrow cursor control once to get the cursor over the 1 of the 10.
- press the right-arrow cursor control 3 times WITHOUT holding down SHIFT at all so the cursor is over the I of INPUT.
- type PRINT
- press RETURN

There – done!

Now, type in a new line 10, again EXACTLY as it is written here, pressing RETURN at the end of the line.

```
10 IF X>5 THEN GOTO 40
```

where the symbol $>$ is taken to mean is bigger than – just in case you haven't met it yet (there is more about it in the section called Mathematical Symbols). Now let's pretend that you got that line wrong in your program, and that what you really wanted was –

10 IF X<1400 THEN GOTO 400

– the symbol < means is smaller than . OK, so how do we change our line 10 to the new line 10?

- hold down SHIFT and press the up-arrow cursor once to put the cursor over the 1 of 10.
- without holding down SHIFT at all, press the right-arrow cursor 7 times to put the cursor over the > sign.
- type the correct symbol – the < sign.
- type 14 – but wait! There aren't enough spaces! So far your line should look like this:

10 IF X<14THEN GOTO 40

How do we insert a few spaces? At the top right of your keyboard, you will find a key marked "INST/DEL".

The word "INST" stands for INSERT. Having found the key, follow these instructions and watch what happens:

- hold down SHIFT and press INST 3 times.
- type 00 and one space with the space bar (NOT with the cursor control key) so that the cursor is now over the T of THEN.
- press the right-arrow cursor control key 12 times, so the cursor is just AFTER the 0 of 40.
- type in another 0.
- press RETURN.

— Done! The reason you were told to use the space bar instead of the cursor control key above to put a space between the 1400 and the THEN is because if you didn't do that, the computer would have become confused. It would have put a strange symbol (which means cursor key to the computer) instead of a space into that position. Don't try messing about with that now! We've more editing to do!

At the moment, our line looks like this:

```
10 IF X<1400 THEN GOTO 400
```

Let's suppose that that STILL isn't exactly what you want. Suppose that you want the following line instead:

```
10 IF X<14 THEN GOTO 400
```

We have to change the 1400 into a 14. This involves deleting or taking out the two zeros. We do this using the DEL part of the INST/DEL key:

- hold down SHIFT and press the cursor up-arrow key once to put the cursor over the 1 of 10.
- press the cursor right-arrow key 12 times (WITHOUT holding down SHIFT) so that the cursor is between the 1400 and the THEN .
- press the INST/DEL key twice (WITHOUT holding down SHIFT).
- press RETURN .

— Done!

Now there is just one other thing to remember. Whenever you've finished editing a program, you must move the cursor onto a blank line before typing anything else like RUN or LIST or something. If you don't move the cursor out of the program, your computer will think you want what you type to be inserted into the program—in other words, it will think you're still editing, and that could cause problems!

Here are a couple of exercises for you to try. Possible answers are given on page 111-112 of Solutions.

EXERCISE 1

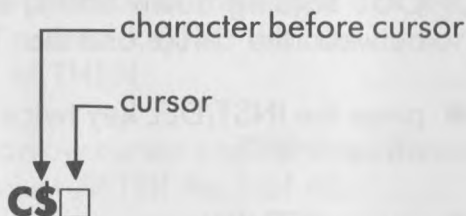
a) Change

10 PRINT C\$

to

10 PRINT X

HINT — remember to get rid of the \$ sign — use either a space from the space bar, or a press of the DEL key — but remember that DELETE always removes the character just BEFORE the cursor, not the one underneath it.



- b) Change

```
10 LET X=20
```

to

```
10 LET D5=20
```

- c) Where must you remember to put the cursor whenever you've finished editing?

END

This statement is very like the STOP statement (see STOP for more about that). It tells your computer to stop running a program at this line. The main difference between END and STOP is that when you use END, the computer does not come up with the "?BREAK IN LINE . . ." message.

An example of an END statement is given below:

```
10 INPUT A$  
20 IF A$="HELLO" THEN GOTO 50  
30 PRINT "GOODBYE"  
40 END  
50 PRINT "HI THERE!"
```

If, when the question mark comes up, you type in "HELLO", your computer will reply with "HI THERE!". If you type in anything else, it will say "GOODBYE" and END the program at line 40 so as not to go on to line 50 and PRINT "HI THERE!" when it's not supposed to.

FOR...NEXT

This structure gets your computer to repeat a part of a program a set number of times.

Suppose you want your computer to do something 10 times — print out "SN7" for example. One way to do this is by means of a program like the one below. Try this by typing it in; don't forget to press RETURN at the end of each line.

```
10 FOR X = 1 TO 10
20 PRINT "SN7"
30 NEXT X
```

Now type RUN and press RETURN. Your computer will put ten SN7's onto the display. It's line 10 that tells the computer how many times to PRINT "SN7". Try changing line 10 to read:

```
10 FOR X = 1 TO 15
```

How many times will the program print now? It will, of course, do the PRINT 15 times — giving you 15 SN7's — what a thought! So then, it's the number after the TO that says how many times to PRINT. A program such as the above is said to loop between lines 10 and 30.

Now try modifying the program by adding:

```
40 PRINT "ALEAT"
```

Now when you run it, the program will still only print one "ALEAT". That's because the loop's contents are the lines between the FOR and the NEXT,

The Loop { 10 FOR X = 1 TO 15
20 PRINT "SN7" ← The loop contents
30 NEXT X
40 PRINT "ALEAT"

EXERCISE 2

As an exercise for yourself, try to make the loop print out "ALEAT" after each "SN7". A solution to this exercise is given on page 113.

As well as counting up to a set number, the loop can be set to count a variable number of times by replacing the counting number by a variable such as L. Try

```
5 LET L = 20
10 FOR X = 1 TO L
20 PRINT "SN7"
30 NEXT X
```

When this is run, the program will execute the loop 20 times.

EXERCISE 3

Modify the above program so that it prints "SN7" 12 times. A solution is given on page 113.

EXERCISE 4

If you have already covered INPUT, modify the above program so that it asks how many times to print SN7 and then does as it's told. Solution is on page 113.

As well as printing messages, a loop can be used to count; try:

```
5 LET L = 20
10 FOR X = 1 TO L
20 PRINT X
30 NEXT X
```

In this case, the program is printing out the value of the loop constant X as it goes along. This can be of use, for example, in printing out, say, the eight times table. To do this it's only necessary to multiply the counting variable X by eight (if you haven't done multiplication yet, check it out under the heading mathematical symbols).

```
5 LET L = 12
10 FOR X = 1 TO L
20 PRINT X*8
30 NEXT X
```

EXERCISE 5

Modify the above program so that it prints out the nine times table. A solution is given on page 114.

Another command used in conjunction with a FOR . . . NEXT loop is STEP; for more details of this see the entry under STEP.

GOTO

This command tells the computer to jump to a new program line and carry on running through the program from there.

For example, we can say:

```
100 LET X = 25
110 GOTO 130
120 STOP
130 PRINT X
140 GOTO 120
```

When your computer runs this, it will do the program lines in this order: 100, 110, 130, 140, 120, rather than 100, 110, 120, 130, 140. This is because, in the example above, when your computer gets to line 110, it is told to GOTO (that is, go to) line 130, which it does right away, missing out line 120. Again, when it reaches line 140, it is told to go to line 120 which it will do, and where it will STOP running the program and will come up with a message “?Break in line 120” – which just means it has been told to stop at line 120.

The word GOTO is useful when used with IF . . . THEN statements to tell the computer what to do after it has made a decision about something (see IF . . . THEN). A good example of this use of GOTO is given in Chapter 5, where Aleat makes up a number guessing game.

INPUT

This command gets your computer to stop a program and accept some information from the user like a number, or a message. When you've told it what it wants to know and have pressed RETURN, it carries on with the rest of the program. Try it with :

```
10 INPUT A$
```

```
20 PRINT A$
```

If you now type RUN and press RETURN, the following happens:

- Computer sees the statement INPUT and prints a question mark (?) on the screen – it's waiting for a message
- Enter something like "GREETINGS EARTHLING"
- Press RETURN
- Computer stores this message in memory location A\$ (see LET for more information on this)
- Computer sees the statement 'PRINT A\$' and then it –
 - looks in memory location A\$ and extracts the message "GREETINGS EARTHLING"
 - PRINTs the message on to the screen
- Computer looks for next program line and when it finds nothing it ends

Memory locations like A\$ which end with a \$ symbol are for storing strings (messages). If you wanted to store a number, you would use a memory location like A or AA or X or KW instead of A\$. You can make up your own memory location names — see LET for more details. (In Chapter 1 Dr W and Aleat put information into memory locations R and H in the program CREWDATA.

INT()

This command tells your computer to take a decimal number and convert it to a whole number, or what a mathematician would call an integer. Positive numbers like 8.1 and 9.64 simply get the figures after the decimal point taken away, in this case we would end up with 8 and 9. Negative numbers like -10.2 and -12.812 have the figures after the decimal point taken away as before, but the whole number is changed to the next more negative number : in our example -10.2 would become -11 and -12.812 would become -13.

Here are some examples of how to use INT(): type each one and press RETURN. Note that INT() cannot be used on its own — it must always be used with another command like PRINT or LET, for example. Just saying INT(2.47) will not work because you haven't told your computer what to do with the answer!

```
PRINT INT(891.2148)
```

this gives 891.

```
10 LET X=INT(-16.5)  
20 PRINT X
```

when you type RUN and press RETURN, you will get a -17 put onto the display.

```
10 LET TP=6+INT(3.1)
20 PRINT TP
```

typing RUN and pressing RETURN will give you a 9.

Can you see why?

LEN()

This is short for LENgth and will give you the length of a message.

```
LEN ("BANANAS") = 7
```

because there are 7 letters in the word bananas.

```
IF A$ = "SN7"
  LEN (A$) = 3
```

LET and MEMORY LOCATION

Have a look at this short program.

```
10 LET X=9
20 PRINT X
```

The line numbers are not too important but the second one must be larger than the first.

Dr W and Aleat worked out that LET means put something into a memory location which, in a way, is like a small box that you can put numbers in.

So if you now say:

```
LET X = 8
```

the computer would put an 8 into the box called X.

We'd better look at the whole process from the beginning! First the LET X bit of the statement, this tells the computer:

"Look for a box labeled X and if you find one put into it the value that follows."

What about when there isn't a box called X?

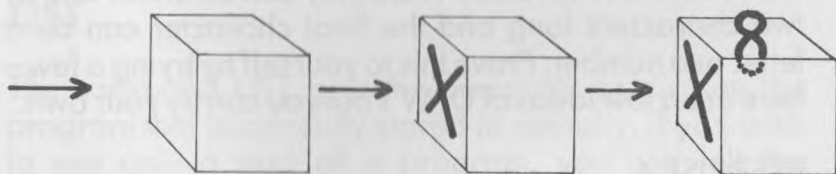
Well, then the statement means:

"Find an unlabeled box, LET is called X and then put a number in it."

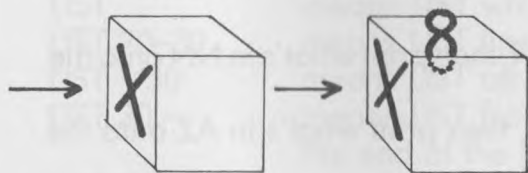
Of course, the number that's used is the one after the equals sign.

Let's look at that in action.

1. When X doesn't already exist.



2. When X already exists.



Once the number is stored in a box, getting it onto a screen is easy – just type in something like :

PRINT X

What it means is:

“Look for a box called X, see what’s in it and then PRINT this value onto the screen.”

So far we’ve called all the boxes X and that’s a bit boring. Unfortunately, most memory location names in BASIC are a little bit that way! Have a look at a few:

- X – boring
- A – perhaps not so boring
- B5 – nearly OK
- ET – that’s better!

What do you notice about them all? Yes, they all begin with a letter. What else? Well, they can be either one or two characters long and the final character can be a letter or a number. Prove this to yourself by trying a few – here are a few ideas of Dr W’s but you can try your own.

EXERCISE 6

- a) Store 3 in D5, then print what’s in D5 onto the screen
- b) Store 11 in NN, then print what’s in NN onto the screen
- c) Store 5 in AZ, then print what’s in AZ onto the screen
- d) Store 21 in Z1, then print what’s in Z1 onto the screen

If you get stuck, see page 114 for the answers.

Just simple numbers are no problem to your Commodore 64, and if you remember in the CREWDATA program Dr W noticed the line:

LET T = R+H

Clearly this involves three boxes: one each called T, R, and H. Can you say in words what you think that means? Think . . . Think. Have you thought? What it acutally means is:

"take the number in box R, add it to the number in box H, and put the answer into box T."

Did you get it right?

Computer programmers refer to the box names such as R, H, and T as VARIABLES.

LIST

The command LIST gets the computer to display the program that is currently stored in memory. If you wish to see only a part of a program, you can tell the computer in detail which lines you want listed:

LIST	means LIST whole program
LIST 20-30	means LIST lines 20 to 30
LIST - 30	means LIST up to line 30
LIST 30 -	means LIST from line 30 to the end of the program.
LIST 30	means LIST only line 30

LOAD

This gets your computer to read a program from a tape and put it into memory.

First of all, you should make sure that your computer and tape recorder or disk drive are connected properly. You should refer to the user manual supplied with the cassette recorder or disk drive.

TAPE

When you type **LOAD** and press **RETURN**, the message

PRESS PLAY ON TAPE

will appear on the screen; if you do as requested the screen will go blank for about 30 seconds, then it will come up with:

OK

SEARCHING

FOUND (then a program name, like **CREWDATA**)

and then it will go blank again for a while – this means it is loading the program. When it is loaded, the screen will report:

READY

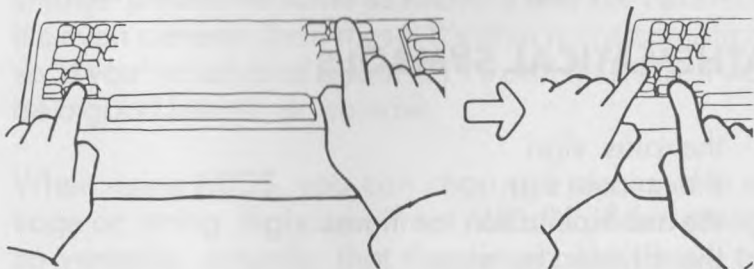
The program will then be ready to RUN (make sure the tape has stopped). Note that your computer will have loaded the first program it found on the tape. If you want it to load a particular program, then you should type in the full name of the program, for example,

LOAD "CREWDATA"

From then on, the instructions are similar to those above, except that your computer will only load the program you've named, even if it isn't the first one on the tape. In our example, it will only load "CREWDATA" even if it finds other programs first. Don't worry if it does find other programs first, it will carry on searching until it finds the one you've asked for.

There is another way to load programs from tape.

Make sure that the screen is clear by holding down SHIFT while pressing CLR/HOME, or make sure you are at the beginning of a new line by pressing RETURN.



Hold down SHIFT and press RUN/STOP. The screen will go blank for a while, and then will come up with

LOAD

SEARCHING

FOUND (and the name of the program)

then the screen will go blank again while your computer reads in (that is loads) the program. When it has done this, it will RUN the program right away — you don't have to type RUN.

DISK

You must specify a program name to load a program from the disk drive. The form is

LOAD "PROGRAM NAME", 8

where PROGRAM NAME is the name of the program you want and ",8" informs the computer to use the disk drive. The disk drive light will go on; when the program is loaded, the screen will report READY.

To store any programs that you write see SAVE.

MATHEMATICAL SYMBOLS

- + the plus sign
- the minus sign
- * the multiplication or times sign
- / the divided by sign
- > the is bigger than sign
- < the is smaller than sign

See IF... THEN and Chapter 5 for how to use these.

These are used in just the same way as when doing sums. Just to try this out, try to get your computer to do the following:

- (i) Add 3 to 4
- (ii) Take 7 from 10
- (iii) Multiply 4 by 3
- (iv) Divide 12 by 3
- (v) If 10 is greater than 3 print YES
- (vi) If 2 is smaller than 21 print NO

Some possible answers to these problems are given below but do try them first!

- (i) PRINT 3+4
- (ii) PRINT 10-7
- (iii) PRINT 4*3
- (iv) PRINT 12/3
- (v) IF 10>3 THEN PRINT "YES"
- (vi) IF 2<21 THEN PRINT "NO"

MID\$()

This is a command that enables you to chop strings into smaller pieces the same as RIGHT\$ and LEFT\$ although it's even cleverer than these. It's also more complicated so, if you've not read about LEFT\$ and RIGHT\$, it would be a good idea to do so now.

When using MID\$, you can chop out pieces of a message or string that come from MIDDLE of that string. It's so versatile, actually, that the pieces don't have to be exactly in the middle, they can be anywhere at all inside the string. Because of this the command MID\$ needs to be told where the piece starts and how many letters to take. Try an example, take:

Where to start
 MID\$("COMPUTER",4,3)
 How many letters to take

COMPUTER
 1 2 3 4
 1 2 3

MID\$("COMPUTER",3,4) = PUT

Just as with LEFT\$ and RIGHT\$, the string can be stored as a variable and the variable name placed in the brackets. Thus, exactly the same effect as above is achieved by:

LET A\$ = "COMPUTER"

then MID\$ (A\$,4,3) = PUT

The rule or syntax of MID\$ is:

MID\$ (string, starting character position, length of string to be cut out)

Once a piece has been cut out of a string it becomes a new string in its own right:

LET B\$ = MID\$ ("COMPUTER",4,3)

would create a variable called B\$ which contains the string PUT.

Earlier on, I said that the command could take characters from anywhere within a string and indeed it can. Take the example:

```
B$ = MID$("COMPUTER",1,4)
```

this would set B\$ to COMP just as if we had used LEFT\$("COMPUTER",4). Try for yourself to strip off the RIGHT-most four characters using MID\$. If you have problems, look at the exercises below and their solutions.

Try a few examples for yourself, using MID\$. The solutions are given on page 114.

EXERCISE 7

MAKE B\$ = "UTER"	from "COMPUTER"
MAKE B\$ = "MAN"	from "COMMAND"
MAKE B\$ = "IN"	from "STRING"
MAKE B\$ = "MOD"	from "COMMODORE"
MAKE B\$ = "VISION"	from "TELEVISION"
MAKE B\$ = "BRIEF"	from "BRIEFCASE"

Once you can use the string handling commands, LEFT\$, RIGHT\$, and MID\$, you can start to make up games where the player has to guess what letters are in a word. Why not have a go?

MULTI-STATEMENT LINES

You don't have to put each new instruction to your computer on a different line. Your Commodore 64 will allow you to put more than one statement on a line instead. You do this by putting a colon (a ':' mark) in between the statements. For example, you've seen programs like this, with one statement per line:

```
10 INPUT A$
20 FOR X=1 TO 10
30 PRINT A$
40 NEXT X
```

However, it could be written like this:

```
10 INPUT A$
20 FOR X=1 TO 10 : PRINT A$
30 NEXT X
```

↑
colon

Or even like this –

```
10 INPUT A$ : FOR X=1 TO 10 : PRINT A$ : NEXT X
```

↑ ↑ ↑
└────────── colon ─────────┘

Usually it is easier to write a program with just one statement per line.

NEW

This command tells the computer to empty out all of its memory locations, and to forget any program it may already have in memory. The command is usually used before you begin writing or loading a new program.

PRINT

Dr W and Aleat found that the word PRINT tells the computer to write something onto the display. For example, when Aleat typed:

PRINT "I LIKE SPACEBURGERS"

and pressed RETURN, the computer put I LIKE SPACEBURGERS onto the display. From this, they worked out that when using PRINT, the computer will put anything that is between quotation marks (the "" marks) onto the display, exactly as it is written. Try it yourself, with any old message you like. For example:-

PRINT "ANY OLD MESSAGE YOU LIKE"

or

PRINT "THIS COMPUTER IS FUNKY"

Remember that you must press RETURN each time you've finished a line to let the computer know you're ready for it to do as you've asked.

Dr W and Aleat discovered that the word PRINT will get the computer to print things other than messages. When they typed (without quotation marks):

PRINT R

the computer wrote out a number. This number was the number which was stored in the memory location called R. (See LET for more on memory locations.) (In the program CREWDATA, location R was used to store the number of robots, although of course it may be used to store other numbers instead.)

You may have noticed in program DRINKSCODE of Chapter 3 that line 40 said:

PRINT "THE CODE WORD";

The last thing on that line is a semi-colon (the ; mark). What this does is tell the computer that the next thing to be put onto the display is to be placed right next to the last thing that was put there. That is, the two things will be printed on the SAME line. For example, if we write:

```
100 PRINT "THIS IS A MESSAGE"  
110 PRINT "FROM YOUR COMPUTER"
```

and then type RUN and press RETURN, the computer will put the following onto the display:

```
THIS IS A MESSAGE  
FROM YOUR COMPUTER
```

If, instead we write (don't forget the semi-colon):

```
100 PRINT "THIS IS A MESSAGE";  
110 PRINT "FROM YOUR COMPUTER"
```

and then type RUN and press RETURN, the computer will write out:

THIS IS A MESSAGEFROM YOUR COMPUTER

It's still not quite right, is it? Do you know why? It's because the semi-colon causes something to be PRINTed RIGHT NEXT to the last thing PRINTed. If we want a space between the words, then we must write it into our PRINT statement:

```
100 PRINT "THIS IS A MESSAGE";  
110 PRINT "FROM YOUR COMPUTER"
```

↑ _____ space

Now, when we type RUN and press RETURN, we get

THIS IS A MESSAGE FROM YOUR COMPUTER

You should try some messages of your own. If you try a message which is too long for one line, your computer will automatically continue the message onto the next line. If your message is much too long, your computer will say: "?SYNTAX ERROR".

Going back to the program DRINKSCODE in Chapter 3, the first instruction after the semi-colon was

```
50 INPUT C$
```


Now, when you tell the computer what message to store in memory location C\$, it always puts that message onto the display as you type it in. In this case however, it puts it up on the SAME line as the words THE CODE WORD, because the PRINT instruction on line 40 ends with a semi-colon.

It is important to note that the semi-colon affects ONLY the position of the very next writing to be put onto the display. Also, it doesn't matter whether that writing is from a PRINT statement or, as above, from an INPUT. It doesn't matter how many other lines of program are between the line with the semi-colon and the line which next writes something onto the display.

Now for some examples (you should try to make up some of your own as well).

```
40 PRINT "HELLO ";  
50 INPUT N$
```

type this in, type RUN and press RETURN. Then type in your name. You will get:

```
HELLO?<your name>
```

You could also try this:

```
40 PRINT "WHAT IS YOUR NAME";  
50 INPUT N$  
60 PRINT "HELLO ";  
70 PRINT N$
```

or

```
40 PRINT "WHAT IS YOUR NAME"  
50 INPUT N$  
60 PRINT "HELLO ";  
70 PRINT N$
```

You should also try these examples without semi-colons to see what happens.

REM

This statement allows you to add comments or REMarks to your program. This is useful in a complicated program when you might forget what it was supposed to do. For instance –

```
10 REM THIS PROGRAM PRINTS A  
20 REM NUMBER ONTO THE SCREEN  
30 INPUT X  
40 PRINT X
```

Your computer will completely ignore any lines which begin with a REM. The REMs are there as REMinders to you – your computer is told what to do on the other lines of the program. It is important to realise that your computer does absolutely nothing with REM statements – it doesn't even display the message after the word REM. To see the REMinder, you have to type LIST (see LIST). Then the REMarks you've placed in the program will be listed along with the rest of the program.

RND()

This tells your computer to think of, that is make up, a number. Normally your computer will make up decimal numbers between 0.0 and 1.0 (or very very near to 1.0 but never quite there). For example, if we write:

```
PRINT RND(0)
```

we will get a RANDOM number — one the computer has just made up — printed onto the display. The zero is not too important at this stage, but the computer does need a number in the brackets though. You can put any positive number you like in these brackets, but a zero gets the computer to make up new numbers. If you use a positive number like 1 or 200 or something, you might find that the computer will always make up the same sequence of numbers when you first start a program. It is usually best to use a zero.

Here are some more examples of how to use RND.

The statement below will give a random decimal number between 0.0 and 100.0. The number 100 after the * sign tells your computer to go up to 100 with its random number.

```
PRINT RND(0)*100
```

The next statement puts a random decimal number between 0.0 and 143.0 into memory location X.

```
LET X = RND(0)*143
```

To get a random whole number between 0 and 30 (see

INT for more about whole numbers) use:

```
PRINT INT(RND(0)*30)
```

Finally, to get a random whole number between 10 and 40 use:

```
PRINT INT(RND(0)*30)+10
```

What your computer does is first of all make up a whole number between 0 and 30, and then it adds 10 to it and puts the answer onto the display.

SAVE

This command tells your computer to store a program onto a tape or disk for later use (see LOAD for reading a program). You don't need to SAVE a program which your computer has loaded, as the original program on the tape or disk is not wiped out by the computer when it is LOADED. You should only need to SAVE a new program.

To store a program on tape or disk, first of all, you must have a program in your computer! Connect your tape recorder or disk drive according to the instructions in the manual that came with the storage device.

TAPE

To make sure that you will not be recording your latest program over one of your older programs, wind the tape to a blank (unrecorded) section. You should think up a name for your program, say "NAME", and type

```
SAVE "NAME"
```

and press RETURN. The screen will display the message:

PRESS RECORD & PLAY ON TAPE

You should do this and press down both the RECORD button and the PLAY button on your tape recorder together. The screen will go blank while the program is being recorded, then it will display the message:

OK
SAVING NAME
READY

this means it has finished. You may still RUN the program, as it remains in the computer's memory after you've saved it. Your computer has only copied it onto the tape, it hasn't removed it from its memory. If you want to check that the program has SAVED correctly you should use the VERIFY command.

DISK

You must format a blank disk before saving programs on it. Insert a blank disk into the disk drive and type

OPEN 15,8,15
PRINT #15,"N0: DISK NAME,ID "

↑
Your Name of the Disk

When the disk drive light goes out, type

CLOSE 15

You may now save a program on the disk with

SAVE "NAME",8

SEMI-COLON

A sign (;) used to separate items. For example,
10 PRINT "MY NAME IS"; A\$

STEP

This command is used in conjunction with the FOR ... NEXT command and has the effect of changing the counting variable in STEPs rather than in ones. The size of the step is defined by the number that follows the STEP command. In the FOR ... NEXT examples a times-table was produced by multiplying the counting variable, but the same effect can be achieved by use of STEP. For example, multiples of 2 can be printed out by:

```
10 FOR X = 2 TO 24 STEP 2
20 PRINT X
30 NEXT X
```

You can investigate the effect of STEP by varying the number after the STEP command in the above program and looking at what happens.

STEP also counts backwards when the number following STEP is negative. Let's get the program to count down from 10 to 1:

```
10 FOR X = 10 TO 1 STEP -1
20 PRINT X
30 NEXT X
```

EXERCISE 8

Modify the above program to make it count down from 20 to 0 in steps of 2. A solution is given on page 115.

STOP

This word, which may only be used within a program, tells your computer to stop running that program when it meets the STOP command.

For example:

```
100 LET X = 5
110 LET Y = 100
120 PRINT X+Y
130 STOP
140 PRINT Y-X
```

When your computer reads line 130, it will stop running the program, and display the message:

? BREAK IN LINE 130

This means that it has stopped because it has been told to STOP on line 130 and to ignore anything that comes after this line.

The word STOP is most useful when you want the computer to stop running a program at a line other than the last line. One example of this is given in the explanation of IF . . . THEN, and another example is given in the section about GOTO.

STRING

This is a term that is used by computer programmers to describe a number of characters that are used together, the sort of thing that we have referred to as "messages".

Thus, in the statement:

```
PRINT "COMPUTER"
```

the "COMPUTER" part is known as a string. It could also be stored as a variable, say A\$ and it is pronounced "A string."

Generally speaking, the term string is only used when letters are present among the characters but a string could be made up of just letters, just numbers or a mixture. In this latter case, we would describe it as an alphanumeric string.

VARIABLE

See LET, page 88.

VERIFY

To make sure that the program has recorded properly, we use the word VERIFY. First of all the program in the computer must be the same one as is on the tape or disk and so you should use VERIFY right after a program has been saved, and before doing anything else.

TAPE

You must make sure that the tape is rewound to a position a little way before the program starts, or back to the beginning of the tape. Then type (if your program is called NAME, for example)

VERIFY "NAME"

and then press RETURN. The screen will display the message:

PRESS PLAY ON TAPE

when you do that, the screen will go blank and shortly will display:

OK

SEARCHING FOR NAME
FOUND NAME

then it will blank out again while it checks the recording to see that it is alright. When that has been done the screen will display the message:

VERIFYING
OK
READY

which means that the recording was OK. If the recording wasn't OK it will report:

? VERIFY ERROR

is which case you should try saving the program again.

DISK

To verify a program that has just been saved on a disk, type

VERIFY "NAME",8

and press RETURN.

The screen will display

SEARCHING FOR NAME
VERIFYING
OK
READY

If you see

?VERIFY ERROR

you should try saving the program again under a different name.

Solutions

EXERCISE 1

a) One possible way to do this (there are many ways, all OK as long as they work!)

- first put the cursor over the 1 of 10 using (for example) the SHIFT and cursor-up arrow keys.
- move the cursor along until it's just past the \$ sign:

10 PRINT C\$ ■

cursor ↑

- press the DEL key twice (don't press SHIFT at all here).
- type X
- press RETURN

b) This can be done like this:

- put the cursor over the 1 of 10 by using SHIFT and the cursor control keys.
- move the cursor until it is on the = sign.
- hold down SHIFT and press INST/DEL once.
- move the cursor back ONE space using the SHIFT and cursor-left arrow key – NOTICE the funny symbol – don't worry! Notice also that the cursor hasn't in fact gone back – it's still over the equals sign.
- NOW move the cursor back two spaces using the SHIFT and cursor-left arrow keys.
- type D5
- press RETURN

– Complicated, isn't it? Try this example a few times until you understand it. If you can't, don't worry – you can always completely type in a new line whenever you've made a mistake – editing is very difficult!

c) Whenever you've finished editing and have pressed RETURN, the cursor must be put in a blank part of the screen – well away from any program lines. That usually means underneath your program – putting the cursor to the right of a short line will not do!

EXERCISE 2

One way to print out Aleat after each SN7 is to do this:

```
10 FOR X=1 TO 15
20 PRINT "SN7"
20 PRINT "ALEAT"
40 NEXT X
```

Now it will print an Aleat each time it goes around the loop because the command PRINT "ALEAT" is inside the loop.

EXERCISE 3

Probably the best way to make the computer execute the loop only 12 times instead of 20 times is to change line 5:

```
5 LET L=12
```

then the rest of the program can stay the same:

```
10 FOR X=1 TO L
20 PRINT "SN7"
30 NEXT X
```

EXERCISE 4

Again, all you have to alter is line 5. The rest of the program can stay the same.

```
5 INPUT X
```

EXERCISE 5

To do the nine-times table, only line 20 has to be changed to:

```
20 PRINT X*9
```

EXERCISE 6

These short programs will do the job: type them in one at a time, pressing RETURN at the end of each line, and typing RUN and pressing RETURN again when you've finished each program.

- a) 10 LET D5 = 3
20 PRINT D5
- b) 10 LET NN = 11
20 PRINT NN
- c) 10 LET AZ = 5
20 PRINT AZ
- d) 10 LET Z1 = 21
20 PRINT Z1

EXERCISE 7

This program will do the job:

```
B$ = MID$("COMPUTER",5,4)
B$ = MID$("COMMAND",4,3)
B$ = MID$("STRING",4,2)
B$ = MID$("COMMODORE",4,3)
B$ = MID$("TELEVISION",5,6)
B$ = MID$("BRIEFCASE",1,5)
```

EXERCISE 8

The following is the modified program:

```
10 FOR X=20 TO 0 STEP -2
20 PRINT X
30 NEXT X
```

Index

- Add(+) 38
- Adding Two Numbers Program 20,21,23
- Addition 12
- ASC() 64
- Bigger Than (>) 43
- Captain Janes' Manual 33
- Cassette loading 5
- CHR\$() 66
- Clearing the screen 13
- CLR/HOME 13
- Code, ASC() 61
- Coding 61
- Colon (:) 76, 98
- Control panel 1
- \$ 28
- Divide (/) 38
- Dollar Sign (\$) 28
- Drinkscod 25, 28
- FOR-TO-NEXT 50
(STEP 51)
- GOTO 28
- Greater than (>) 43
- Guessing Game 41, 42, 46
- Guessing Game Strategy (Dr W's) 47
- HOME 13

IF-THEN 29

INPUT 9

INT() 36

Integer 36, 37

Is bigger than (>) 43

Is smaller than (<) 44

Keyboard 2

Keys 3

LEFT\$ 53

LEN() 88

Less than (<) 44

LET 12

Line No. 15

LIST 15, 16

Loading the cassette for the
first time 5

Location in memory 11

Loop 82

Mathematical Symbols

\$ 28

+ 38

- 38

* 38

/ 38

> 43

< 44

" " 9

Memory Location 11

Messages 28

MID\$ 55

Multiply (*) 38

Multi-Statement Lines 98

Name recognition program 29, 30, 32

Names for memory location 11

NEW 19

NEXT 50

Number guessing game 41, 42, 46

Numbers and memory location 13

Old School, the 56

Order of line no.'s 17

Oxygen 59

Power-up 1

PRINT 9

PRINT with "" 12

Program to add two numbers 20, 21, 23

Program to recognize a name 29, 39, 32

Random numbers 34

Random whole numbers 37, 38

RIGHT\$ 53

RND() 36

RUN 18

Semi-colon (;) 106

Shift 4

Smaller than (<) 44

STEP 51

Storage locations 11

Strings 28

Subtract (-) 38

Switching on 1

THEN 29

TO 50

Upper symbols on keys 4

Variable 91

Variable names 11

A revolutionary FUN approach to computer learning

- *Race against time to escape from an alien planet by applying your understanding of BASIC to control a spaceship computer.*
- *Discover the wonders of BASIC programming using your Commodore 64.*

You are the hero of this mission in space. By some strange accident you are transported to a faraway galaxy and join three wanderers. There is Dr. W—a wise, but slightly odd, teacher and inventor; Aleat—a beautiful 240-year-old Arfonite; and her trusty, rusty robot companion SN7. Aboard a spaceship long ago deserted by its crew, you find a computer remarkably similar to your Commodore 64. If you can make the computer understand your instructions you can escape. If you can't operate the computer, you'll remain **LOST IN SPACE**.

The *software* inside this package is specially designed to teach the fundamentals of BASIC programming, and it contains programs from the spaceship computer.

Dr. W also carefully explains each BASIC command in an *easy-to-understand* section at the end of the book.



HAYDEN BOOK COMPANY

a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

ISBN 0-8104-7633-9